Universidad Nacional Mayor de San Marcos
Universidad del Perú. Decana de América
Facultad de Ingeniería Electrónica y Eléctrica
Escuela Profesional de Ingeniería de Telecomunicaciones

Diseño y simulación de una red definida por software para la implementación de un laboratorio avanzado de datos para la EP de Telecomunicaciones de la Facultad de Ingeniería Electrónica y Eléctrica de la Universidad Nacional Mayor de San Marcos

TESIS

Para optar el Título Profesional de Ingeniero de Telecomunicaciones

AUTOR

Ernesto RODRÍGUEZ GUERRERO

ASESOR

Ing. Daniel DÍAZ ATAUCURI

Lima, Perú

2020

Referencia bibliográfica

Rodríguez, E. (2020). *Diseño y simulación de una red definida por software para la implementación de un laboratorio avanzado de datos para la EP de Telecomunicaciones de la Facultad de Ingeniería Electrónica y Eléctrica de la Universidad Nacional Mayor de San Marcos.* Tesis para optar el título de Ingeniero de Telecomunicaciones. Escuela Profesional de Ingeniería de Telecomunicaciones, Facultad de Ingeniería Electrónica y Eléctrica, Universidad Nacional Mayor de San Marcos, Lima, Perú.

Hoja de metadatos complementarios

Código ORCID del autor	"—"
DNI o pasaporte del autor	72730688
Código ORCID del asesor	Orcid.org/0000-0001-5747-2795
DNI o pasaporte del asesor	07139361
Grupo de investigación	Grupo de Aplicaciones de las Tecnologías de Información y Comunicación de la Facultad de Ingeniería Electrónica y Eléctrica de la UNMSM
Agencia financiadora	a) País: Perú b) Agencia financiadora: Vicerrectorado de Investigación y Postgrado UNMSM - VRIP c) Nombre del programa financiero: Programa de Promoción de Tesis de Pregrado. d) Número de contrato: Comunicado: N°03-2017 VRIP.
Ubicación geográfica donde se desarrolló la investigación	Lugar: Facultad de Ingeniería Electrónica y Eléctrica de la Universidad Nacional Mayor de San Marcos. Coordenadas geográficas: -12.05516873 S, -77.08709829 O
Año o rango de años en que se realizó la investigación	2017 - 2020
Disciplinas OCDE	Telecomunicaciones: https://purl.org/pe-repo/ocde/ford#2.02.05

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
(Universidad del Perú, DECANA DE AMÉRICA)
FACULTAD DE INGENIERÍA ELECTRÓNICA Y ELÉCTRICA

ACTA DE SUSTENTACIÓN N° 001-EPIT-FIEE/2021

TESIS N° 001-FIEE/2021 PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERÍA DE TELECOMUNICACIONES

Los suscritos Miembros de Jurado, nombrados por la Dirección de la Escuela Profesional de Ingeniería de Telecomunicaciones, reunidos en la fecha bajo la Presidencia del Dr. Santiago Rojas Tuya e integrado por los Ingenieros: Mg. Wilbert Chávez Irazábal, Ing. Milton Rios Julcapoma y el Mg. Daniel Díaz Ataucuri (Miembro -Asesor)

Después de escuchar la Sustentación de Tesis del Bachiller Ernesto Rodríguez Guerrero (11190191), para optar el Título Profesional de Ingeniero de Telecomunicaciones por la modalidad de Titulación Ordinaria, quien expuso su TESIS: "Diseño y Simulación de una Red Definida por Software para la implementación de un laboratorio avanzado de datos para la EP de Telecomunicaciones de la Facultad de Ingeniería Electrónica y Eléctrica de la Universidad Nacional Mayor de San Marcos".

Se acordó **APROBADO** por **UNANIMIDAD**

Con la Nota de **DIECISIETE (17)**

Ciudad Universitaria, 22 de enero de 2021

Dr. Santiago Rojas Tuya
Presidente de Jurado

Mg. Wilbert Chávez Irazábal
Miembro de Jurado

Ing. Milton Rios Julcapoma
Miembro de Jurado

Mg. Daniel Díaz Ataucuri
Miembro de Jurado-Ases

Mg. Carlos Alberto Sotelo Lopez
Director de la EPIT

UNMSM
Firmado digitalmente por UTRILLA SALAZAR Dario FAU 20148092282 soft
Motivo: Soy el autor del documento
Fecha: 28.01.2021 16:16:55 -05:00

Dr. Dario Utrilla Salazar
Decano FIEE – UNMSM

DEDICATORIA

Dedico esta tesis a Dios, a mi padre que está en el cielo, a mi madre que siempre ha sido un apoyo fundamental en mi formación humana y profesional, a mi tío que ha sido como un segundo padre que me ayudó todos los años de mi vida, a toda mi familia y profesores, pues a ellos les debo el desarrollo personal y profesional logrado.

AGRADECIMIENTOS

Agradezco a mi asesor de Tesis Ing. Daniel Díaz Ataucuri por su paciencia y apoyo durante el desarrollo de la Tesis, así como los conocimientos brindados durante mi formación académica.

Agradezco a mi madre y familia por haberme soportado moralmente para continuar mi camino profesional.

INDICE

CONTENIDO

RESUMEN

En la presente tesis, para obtener el título de Ingeniero de Telecomunicaciones, se estudia la arquitectura de una Red Definida por Software-SDN, tanto a nivel de hardware: equipos requeridos y función de cada uno de ellos, como a nivel de Interfaz de Programación de Aplicaciones-API northbound y southbound. Con el fin de proponer una arquitectura de Red Definida por Software-SDN para la implementación de un laboratorio avanzado de datos en la Facultad de Ingeniería Electrónica y Eléctrica-FIEE de la Universidad Nacional Mayor de San Marcos-UNMSM, el cual cuente con la opción de conectividad remota para poder realizar las actividades de laboratorio (tanto evaluaciones como ensayos) a distancia. La propuesta de SDN se sustenta en las simulaciones realizadas en Mininet, previo estudio de las especificaciones de esta arquitectura SDN.

Esta tesis pretende ser el primer paso para una futura implementación de un laboratorio de Red Definida por Software en la Facultad de Ingeniería Electrónica y Eléctrica de la UNMSM y con esto dar el salto tecnológico que permita colocar a nuestra facultad y casa de estudios como referente en investigación a nivel nacional, teniendo un laboratorio que permita preparar a los estudiantes de la universidad acorde con las necesidades actuales de investigación y adaptabilidad al cambio tecnológico.

ABSTRACT

In this thesis, to obtain the title of Telecommunications Engineer, the architecture of a Network Defined by Software-SDN is studied, both at the hardware level: required equipment and function of each of them, as well as at the level of the Application Programming Interfaces-API northbound and southbound. In order to propose an architecture of Software Defined Network-SDN for the implementation of an advanced data laboratory in the Faculty of Electrical and Electronic Engineering-(FIEE in spanish) of the National University of San Marcos-UNMSM. The proposal of SDN is based on the simulations carried out in Mininet, after studying the specifications of this SDN architecture.

This thesis intends to be the first step for a future implementation of a Software Defined Network laboratory in the Faculty of Electronic and Electrical Engineering of the UNMSM and with this, to give the technological leap that allows to place our faculty and house of studies as a reference in research at national level, having a laboratory that allows to prepare the students of the university according to the current needs of research and adaptability to technological change.

CAPÍTULO I

OBJETIVOS DE LA TESIS

1.1 INTRODUCCIÓN

En la actualidad nos encontramos en una constante evolución en las Tecnologías de la Información y Comunicación-TIC, la cual conlleva a cambios en las preferencias y uso de datos en la red por parte de los usuarios, parte de esta evolución es Internet de las Cosas-IoT, el salto de 4G hacia 5G para el 2021, el control centralizado (SDN), virtualización de la infraestructura de la red; así como la virtualización de sus servicios (NFV), entre otros.

Esta evolución del uso de la red por parte de los usuarios demanda a las redes de datos una mayor escalabilidad y flexibilidad, ante el cual tenemos como opción de uso de la Red Definida por Software-SDN que presenta como características básicas ser una red flexible, de gestión centralizada, ágil y programable.

En la Universidad Nacional Mayor de San Marcos, FIEE-UNMSM, EP Ing. de Telecomunicaciones, es de vital importancia el estudio y futura implementación de este tipo de red debido a que permitirá contar con una infraestructura que facilite sus actividades de investigación e innovación; ubicándonos dentro de los principales centros de estudios de la región, para ello se requiere complementar el equipamiento del laboratorio del curso Arquitectura de Redes Virtuales (SDN/NFV), el cual fue aprobado en el Plan de Estudios 2018 y el primer paso es tener una propuesta de laboratorio con estas características.

En este entorno se da la presente tesis para obtener el título profesional de Ingeniero de Telecomunicaciones, como un paso inicial para una implementación futura de un laboratorio SDN en la Facultad de Ingeniería Electrónica y Eléctrica de la UNMSM que permita entender esta tecnología e iniciar una línea de investigación. Para ello se entrega una propuesta de red y el estudio de su funcionamiento en base de la simulación utilizando Mininet.

Para fundamentar la propuesta de laboratorio y el estudio de su funcionamiento se partirá del análisis de los estándares que definen la arquitectura SDN, el estado del arte y la investigación actual sobre esta nueva tecnología de red.

1.2 PLANTEAMIENTO DEL PROBLEMA

Actualmente, en la FIEE-UNMSM, EAP-Ing. De Telecomunicaciones, se encuentra aprobado el Plan de Estudios 2018[1] el cual incluye el curso Arquitectura de Redes Virtuales (SDN/NFV), sin embargo, la facultad aún no cuenta con un laboratorio acorde con este curso, tampoco con un trabajo de investigación que brinde los lineamientos y requisitos necesarios para su implementación, de modo que se requiere una tesis en la cual se realice el diseño y la simulación de la red de laboratorio, que sirva como documento guía necesario para su futura implementación.

También, debido a la coyuntura actual en la que nos encontramos (pandemia debido al COVID-19)[2] es necesario contar con un laboratorio que brinde conectividad remota, para que así los estudiantes que requieran practicar en éste lo puedan hacer desde la seguridad de sus domicilios pues más del 80% de países está experimentando cierres en sus centros de estudios[3] en nuestro caso particular el Estado anunció la distribución de más de 800000 tablets para alumnos y 97000 para profesores[4], del mismo modo se viene dando la entrega de dispositivos informáticos y electrónicos en nuestra casa de estudios[8], todo esto en marco del decreto legislativo Nro 1465-2020[5].

Por lo expuesto, es necesario un diseño y simulación de un laboratorio SDN, para que en un futuro cercano se implemente en la Facultad de Ingeniería Electrónica y Eléctrica de la Universidad Nacional Mayor de San Marcos, a través de inversión por parte de la facultad, tomando como modelo este documento, de manera que posteriormente se comprenda esta tecnología no sólo en la teoría sino también en la práctica y se inicien actividades de investigación en esta rama, colocando a la UNMSM como una universidad líder en ciencia y tecnología en el área de redes o *networking*.

1.3 HIPÓTESIS

1.3.1 Hipótesis General

Con las herramientas de software libre disponibles, se puede realizar el diseño y simulación de una red SDN para educación e investigación en la FIEE-UNMSM.

1.3.2 Hipótesis Específicas

H1: Analizando las especificaciones técnicas de los principales organismos de estandarización que definen la arquitectura SDN se puede diseñar una topología de laboratorio SDN para la FIEE-UNMSM a nivel físico y lógico.

H2: Utilizando contenedores o máquinas virtuales de software libre se puede simular el funcionamiento del controlador open-source que se elegirá en el diseño.

H3: Utilizando herramientas de software libre se puede simular el funcionamiento de la topología de red propuesta de un laboratorio SDN para la Facultad Ingeniería Electrónica y Eléctrica de la Universidad Nacional Mayor de San Marcos comprobando que su tiempo de convergencia es menor respecto a una red legacy.

H4: Habiendo hecho el diseño y simulación de la red de laboratorio SDN para la Facultad Ingeniería Electrónica y Eléctrica de la Universidad Nacional Mayor de San Marcos se puede diseñar la topología complementaria que permitiría el acceso remoto.

1.4 JUSTIFICACIÓN

Además existen disposiciones legales que empiezan a promover la investigación a todo nivel, el desarrollo tecnológico e innovación en las universidades peruanas como por ejemplo la ley 30309 que promueve la investigación científica, desarrollo e innovación tecnológica [6] o la ley 28303 que da el marco de ciencia, tecnología e innovación en el Perú[7]. Con esto, nos encontramos ante un relanzamiento de la investigación e innovación, donde es necesario contar con

los medios tecnológicos para enfrentar estos retos; siendo las redes de datos un medio tecnológico necesario para realizar una investigación multidisciplinaria remota, por lo que nos obliga a entender e innovar ésta nueva red emergente.

Adicionalmente desde el año 2018 el “Perú es aceptado como Miembro Pleno del acuerdo APEC Engineer – Ingeniero APEC”[8] lanzó una “campaña para incentivar la educación técnica profesional en los países miembro”, es decir, los ingenieros peruanos que se logren acreditar como APEC Engineer serán reconocidos y estarán acreditados para ejercer la ingeniería en cualquiera de los 15 países que conforman el acuerdo, entre ellos: Estados Unidos, Canadá, Australia, Japón, entre otros. Lo cual implica que es necesario que nuestros egresados terminen la carrera con una sólida base en las últimas tendencias tecnológicas.

Actualmente, la especialidad de Ingeniería de Telecomunicaciones de la Facultad de Ingeniería Electrónica y Eléctrica ha aprobado la nueva malla curricular, en la cual se encuentran áreas clave como redes de datos que abarca diferentes temas como redes de computadoras, software para telecomunicaciones, redes de banda ancha y arquitecturas virtuales SDN//NFV.

Las grandes empresas que fabrican equipos de telecomunicaciones, como CISCO, impulsan el uso de SDN para acelerar la implementación y la distribución de aplicaciones con un mínimo de inversión en tecnologías de la información (TI) [9]. Las empresas de servicios de Internet como Telefónica, empresas comerciales como HUAWEI y la Universidad Politécnica de Madrid-UPM han realizado una experiencia pionera a nivel mundial, demostrando la aplicación de criptografía cuántica en redes ópticas comerciales y su integración con la operación de la red por medio de tecnologías basadas en SDN (Software Defined Networking) [10].

Para que la FIEE-UNMSM, facultad vinculada con las nuevas tecnologías de las telecomunicaciones, tenga todos los medios para formar a los estudiantes consecuentemente con este gran salto tecnológico, necesita disponer de un laboratorio que fortalezca a sus estudiantes en la tecnología de red de datos e infraestructura de última generación. Siendo una de ellas la Red Definida por

Software-SDN, promovida por la Open Network Foundation[11] como una red programable, ágil, de estándar abierto y de gestión centralizada; esta red será la que facilitará la gestión, configuración y necesidades de los investigadores de manera sencilla a través de interfaces de usuarios.

Es en este contexto la necesidad de que la Facultad de Ingeniería Electrónica y Eléctrica cuente con un laboratorio de nuevas tecnologías para reforzar la enseñanza y fomentar la investigación; de manera especial en redes definidas por software base de las TIC emergentes. Esta propuesta de tesis tiene como objetivo diseñar y simular escenarios SDN para la implementación de un laboratorio SDN para reforzar la formación de los alumnos de la Facultad y fomente la investigación en estos temas de interés.

1.5 ANTECEDENTES

Actualmente se han identificados trabajos de investigación que han generado la titulación de sus autores y elaboración de artículos sustentados en congresos académicos. Estas informaciones serán el punto de inicio en el desarrollo de la presente tesis para obtener el título de Ingeniero de Telecomunicaciones por la UNMSM:

• En el White paper “2020 Global Networking Trends Report (2020)”[12] se muestra que las SDN actualmente transportan el 23% del tráfico dentro de los data centers empresariales y se espera que para el próximo año sea el 44%, además de que cerca del 60% de los líderes de TI tienen ya desplegadas alguna de las formas de SDN en sus centros de datos debido a que esto junto a la virtualización de funciones de red permite una orquestación central de carga con servicios computacionales y de almacenamiento.

• En el documento “EdTech Knowledge Pack on Remote Learning” response to COVID-19 (2020)”[13] se menciona que más del 80% de países han cerrado sus escuelas debido a la pandemia y que una de las herramientas clave de educación a distancia es la educación por internet.

• En el documento “Coping with COVID-19: International higher education in Europe” (2020)”[14] el cual es un estudio acerca de las medidas tomadas por las

instituciones de educación superior en Europa, se menciona que las medidas adoptadas debido a la pandemia han afectado la movilidad foránea del 73% de los estudiantes y la movilidad interna del 48% de estudiantes.

• En el White paper “Cisco Annual Internet Report (2018-2023)”[15] presentado el año 2018 se muestra que las redes definidas por software son necesarias para poder ser parte de la transformación de la industria de las redes de datos, siendo que de acuerdo a la encuesta de la “International Data Corporation (IDC) (2018)” de las 1202 empresas líderes en TI encuestadas el 39.6% ya usaban SDN al momento y el 38.3% lo tenía planificado para los siguientes 12 meses.

• En la tesis o trabajo fin de grado (TFG) para obtener el título de ingeniero informático en la Universidad Politécnica de Madrid “*Despliegue de los laboratorios de SDN virtuales utilizando Open vSwitch*” de Pablo Sayans Cobos (2018, España) [16] se concluye que los modelos de redes *legacy* no están preparados para gestionar la conectividad de las infraestructuras modernas, las cuales han ido cambiando con la llegada de la virtualización y los servicios en la nube, ya que estas redes son resilentes ante fallos o cambios puntuales.

• En el Artículo “*Comparative analysis of SDN and conventional networks using routing protocols”* de Deepthi Gopi, Samuel Cheng y Robert Huck (2017, USA) [17] se concluye que los protocolos de enrutamiento convencionales como RIP, OSPF, EIGRP y BGP tienen un sistema rígido e intrincado que estrechan la adaptabilidad de las redes a la siempre cambiante Internet. La aparición de las Redes Definidas por Software-SDN proporciona una solución a este problema. Debido a la imparcialidad de un controlador centralizado, SDN proporciona un método efectivo en términos de cálculo de enrutamiento y fino control sobre los paquetes de datos. En este artículo se mide el rendimiento analizando el tiempo de convergencia de enrutamiento durante una falla de enlace con respecto a la escala topológica para mostrar que el enrutamiento/reenvío de SDN es mejor comparado con el enrutamiento convencional.

• En el Artículo “*Performance analysis of SDN/OpenFlow controllers: pox versus floodlight”* de Idris Z. Bholebawa y Upena D. Dalal (2017, España) [18], se afirma que las Redes Definidas por Software-SDN representan una arquitectura de red

emergente que es adaptable, dinámica, rentable, y gestionable. Se muestra un estudio comparativo entre los controladores POX y Floodlight, concluyendo que respecto al RTT (*Round-trip Time* o tiempo de ida y vuelta) y el *throughput* o rendimiento en diversas topologías (simple, linear y árbol) el controlador Floodlight entrega un desempeño más eficiente.

• En el Artículo "*Dynamic QOS/QOE assurance in realistic nfv-enabled 5g access networks"* de Jose Juan Pedreno Manresa, Pouria Sayyad Khodashenas, Muhammad Shuaib Siddiqui y Pablo Pavon-Marino (2017, India)[19], se analiza las tecnologías de NFV y SDN y los sistemas de radio habilitados para la nube considerando el agrupamiento de recursos, escalabilidad, interoperabilidad entre capas y eficiencia espectral.

• En el Artículo "*Centflow: centrality-based flow balancing and traffic distribution for higher network utilization"* de Rajesh Challa, Seil Jeon, Dongsoo S.Kim y Hyunseung Choo (2017, USA)[20], se concluye que las Redes de Próxima Generación o NGN, por sus siglas en inglés, están abrazando dos principios esenciales del paradigma de las Redes Definidas por Software, la segregación funcional del plano de control y de reenvío, y la centralización lógica del plano de control. Un control centralizado mejora significativamente la gestión de la red regulando la distribución del tráfico dinámica y efectivamente, además, en el artículo se propone un nuevo algoritmo de ruteo, el CentFlow para un dominio de SDN y aumentar la utilización de la red.

• En el Artículo "*Making powerful friends: introducing onos and net2plan to each other*" de Pontus Sköldström Ćiril Rožić, Jose Juan Pedreno Manresa (2017, España) [21], se concluye que la planificación de red debe realizarse periódicamente para asegurarse que la red cumpla con todas las demandas de tráfico. Sin embargo, esta planificación es de manera individual para cada capa, lo cual conlleva a que se incremente el OPEX y el CAPEX; además, en este artículo se propone la integración de la plataforma Open-Source ONOS con el planificador de red también Open-Source de nombre Net2Plan.

• En el Artículo "*NFV and SDN guide for carriers and service providers"* (2017)[22] publicado por Ciena Corporation, se menciona que los ISP intentan desplegar

SDN y los criterios más prioritarios son: la capacidad de responder más rápidamente a los requerimientos de sus clientes gracias a la agilidad operacional de la red; la reducción del OPEX gracias a la automatización de la red y la reducción de CAPEX gracias a la optimización del uso de recursos tales como ancho de banda y la capacidad de la red de data center.

• En la Tesis de maestría: "*Trusted communication in SDN openflow channel*" de Marika Haapajärvi de la JAMK University of Applied Sciences (2017, Finlandia)[23] se parte de que las redes tradicionales tienen la característica de ser de vendedores cerrados (vendor-locked), propietarios y dispersos especialmente desde el punto de vista de la gestión sin embargo, el remedio para esto se dio hace 20 años en la mente de los investigadores revolucionarios Wetherall, Guttag & Tennenhouse en 1998 mediante las interfaces de programación de aplicaciones las cuales permiten la administración de la red mediante programación en lugar de cambios físicos, característica hoy de SDN.

• En la Tesis de grado "*Prototipo de una SDN utilizando herramientas open-source*" presentada por Juan Francisco Guano Viscarra de la Escuela Politécnica Nacional de Quito (2017, Ecuador)[24], afirma que el principal objetivo de SDN es el unificar la administración de la red y utilizando software controlar la conectividad y flujo de datos, teniendo el plano de control la característica de programabilidad.

• En la Tesis de Maestría "*Analysis of openflow and netconf as sbis in managing the optical link interconnecting data centers in an SDN environment*" presentado por Karpakamurthy Muthukumar de la Universidad de Carleton, Canadá (2016)[25], se considera que la tecnología SDN ha sido inicialmente aplicada a todos los tipos de tamaños de red, desde servicios ethernet hasta largos entornos de nube. Recientemente el interés se ha centrado en extender la programabilidad de las redes de transporte óptico, en la tesis se despliegan los dos mayores protocolos de gestión southbound del ancho de banda bajo demanda el NETCONF y el OpenFlow.

• En la tesis "*Diseño e implementación de un controlador SDN/OpenFlow para una red de campus académica*" presentada por los bachilleres Gabriel Josías

Cuba Espinoza y Juan Manuel Augusto Becerra Avila de la PUCP (2016) [26], se diseña un controlador SDN para la red del campus de la PUCP para una posible implementación futura y se hace un análisis de su escalabilidad y flexibilidad; en base a este análisis selecciona el controlador FloodLight, donde se demuestra que usando este mecanismo se obtiene un 25% de uso en las TCAM de los Switches y en la capacidad del controlador.

• En la tesis o trabajo de fin de master "*Estudio de las redes definidas por software y escenarios virtuales de red orientados al aprendizaje*" de Javier Cano Moreno de la Universidad Politécnica de Madrid (2015) [27], se analiza la evolución vertiginosa de la tecnología y la virtualización y se hace un estudio sobre las diversas plataformas y herramientas que se pueden usar en SDN.

• En la tesis o trabajo fin de grado (TFG) "*Estudio de la factibilidad técnica en la implementación de la tecnología SDN en las redes de datos de área local*" de Xavier Andrés Domínguez Briones. (2015 Universidad Politécnica Salesiana)[28], estudia la factibilidad técnica de la implementación de la tecnología SDN en las redes de área local, en el cual se analizan los procesos que realizan todos los elementos de una red de datos que tenga implementado SDN y el controlador OpenFlow; además de analizar los requerimientos mínimos necesarios para que un equipo de comunicación sea compatible con OpenFlow.

• En la Tesis de Grado "*Opendaylight SDN controller platform*" de Bernat Ribes García de la Universidad Politécnica de Catalunya, España (2015)[29], se afirma que las Redes Definidas por Software se presentan como un nuevo paradigma con el objetivo de simplificar la gestión y creación de red. Las Redes Definidas por Software están basadas en la separación física del plano de control y el plano de reenvío, introduciendo un elemento lógico centralizado, el controlador. Esa tesis documenta la infraestructura de los controladores y proporciona una base sólida para el desarrollo de aplicaciones OpenDaylight.

• En el trabajo de fin de master "*Estudio del estado del arte de la ingeniería de tráfico en redes SDN. caso de estudio Oshi*" de María José Argüello Vélez. (2015) [30] Universidad Politécnica de Madrid, España, se justifica el uso de redes SDN

por ser de gran interés para los ISPs y se hace un análisis sobre ambientes híbridos y servicios capa dos en estos ambientes como VLL y Pseudo Wire.

• En el white paper "*Practical implementation of SDN & NFV in the WAN*", octubre (2013) [31], hecho por Sterling Perrin y Stan Hubbard. Se destaca los beneficios tanto de la aplicación de una red definida por software como de la virtualización de red y de la asociación entre HP y Wind River para mostrar que el desempeño de un Open vSwitch estándar de fuente abierta puede ser mejorado en gran magnitud, fusionando el Kit de Desarrollo de Plano de Datos (DPDK por sus siglas en inglés) de Intel, con el Perfil de Virtualización Abierta (OVP por sus siglas en inglés) de Wind River obteniéndose una ganancia de rendimiento de diez veces en la conmutación de paquetes para habilitar a los proveedores de servicios soportar mayor cantidad de máquinas virtuales.

1.6 OBJETIVOS

1.6.1 Objetivo General:

Diseñar y Simular una Red Definida por Software para proponer la implementación de un laboratorio avanzado de datos en la Facultad de Ingeniería Electrónica y Eléctrica de la Universidad Nacional Mayor de San Marcos probando que ésta es teóricamente más eficiente en gestión y funcionamiento respecto a la legacy.

1.6.2 Objetivos Específicos:

1) Diseñar la topología de laboratorio SDN para la FIEE-UNMSM, incluyendo la descripción de los equipos que debe tener.

2) Simular el funcionamiento del controlador open-source elegido para la red de laboratorio SDN.

3) Simular el funcionamiento de la topología propuesta de laboratorio SDN (controlador + nodos), utilizando herramientas de software libre, para la Facultad Ingeniería Electrónica y Eléctrica de la Universidad Nacional Mayor de San Marcos y revisar los resultados para analizar su mejora en eficiencia.

4) Diseñar la topología que permita conectividad remota al laboratorio SDN para la FIEE-UNMSM.

1.7 METODOLOGÍA

Para el desarrollo del presente trabajo de investigación la metodología utilizada constará de tres partes: la primera recopilar la documentación técnica de la tecnología SDN, la segunda que es el diseño de la topología y finalmente la tercera que será su simulación.

Recopilación de la documentación técnica: El proceso de investigación se inicia realizando un análisis de las especificaciones técnicas, principales tesis de maestrías y doctorales relacionadas a las redes definidas por software.

• Se consulta tesis y artículos sobre modelos y arquitectura de SDN, así como información de la Open Network Fundation quien junto a la IETF son quienes brindan estándares para SDN, para tener mayor conocimiento del protocolo OpenFlow con el cual se puede trabajar en la interfaz southbound en SDN.

• Se consulta la web del software de simulación Mininet, también el manual del software de simulación GNS3 el cual se encuentra en la página web oficial de GNS3 y los manuales de instalación de los controladores SDN Controller, Floodlight y del OpenVSwitch Pica8.

Diseño de la topología**:** Para poder cumplir con los objetivos de la tesis se propondrá una red de laboratorio SDN para la Facultad de Ingeniería Electrónica y Eléctrica de la UNMSM.

Simulación de la topología**:** La simulación se realizará probando los softwares disponibles para controladores, así como el entorno de simulación de Mininet, para evaluar diferencias respecto a una red legacy.

1.8 CONTENIDO DE LA TESIS

En el primer capítulo se describe el planteamiento del problema, la hipótesis, la justificación, los objetivos de la tesis, los antecedentes y la metodología a usarse. En el segundo capítulo se describe el marco teórico donde se analizan los estándares, así como los principales artículos técnicos indexados sobre el tema de la presente tesis. En el tercer capítulo se propone la topología del laboratorio de Red Definida por Software para ser implementada en la Facultad de Ingeniería Electrónica y Eléctrica de la Universidad Nacional Mayor de San Marcos, así como los aspectos económicos de este laboratorio. En el cuarto capítulo se describe la simulación de la topología de laboratorio SDN para la FIEE-UNMSM propuesta, utilizando herramientas de software libre. En el quinto capítulo explican los resultados de la tesis. En el sexto capítulo se dan las conclusiones y recomendaciones. Finalmente se indica la bibliografía y los anexos de la tesis.

CAPÍTULO II

MARCO TEÓRICO

2.1.- INTRODUCCIÓN

Una Red Definida por Software es una arquitectura de red relativamente nueva, que da paso al control centralizado en vez del control distribuido que se tiene en las redes antiguas o *legacy*. En esta arquitectura de red el plano de control contiene al controlador el cual es el cerebro de la red ya que realiza la función de dar las órdenes a la red; en el plano de datos se ubican los dispositivos de interconexión físicos o virtuales que definen la trayectoria de los datos en la red [32]. El modelo SDN está conformado por tres capas: la capa de infraestructura que contiene los dispositivos de interconexión o switches SDN (Switches OpenFlow), la capa de control que contiene el controlador SDN y la capa de aplicación para la gestión de toda la red, como se ilustra en la figura 2.1.

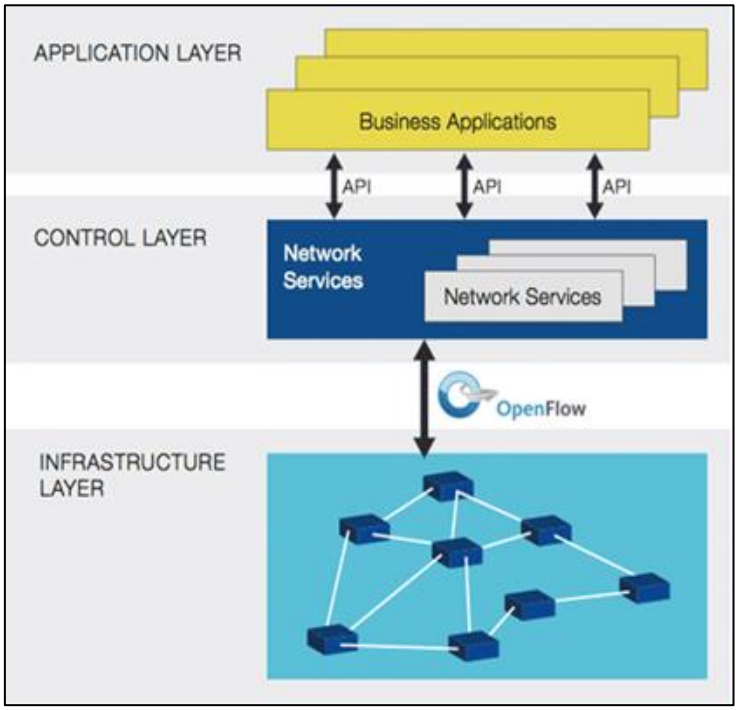


Figura 2.1. Diseño simplificado de una Red Definida por Software. Fuente: Open Networking Foundation. [https://www.opennetworking.org/sdn-resources/sdn-definition]

Existen interfaces de programación de aplicaciones (API por sus siglas en inglés) dirigidas hacia el norte (capa de control hacia la capa de aplicación) y hacia el

sur (capa control hacia la capa de infraestructura). La interfaz hacia el norte o "Northbound Interfaces" (NBIs) es una interfaz entre las aplicaciones de la red definida por software y el controlador (o controladores). La interfaz hacia el sur o "Southbound Interfaces" (SBIs) está compuesta por "*Control to Data-Plane Interface*" (CDPI) que es la interfaz entre el controlador y el plano de datos; la interfaz hacia el sur más utilizada es OpenFlow, aunque existen otras como OVSDB, NETCONF, LISP, BGP, PCEP, SNMP. En esta arquitectura de red, la gestión de la red es realizada desde el controlador hacia los dispositivos de la capa de infraestructura.

La conmutación y/o enrutamiento se realiza manejando tablas de flujo y tablas de grupo entregadas por el controlador, existe una cantidad importante de controladores desarrollados en lenguaje Java y Python que son de código abierto, se puede realizar redundancia conectando más de un controlador a un mismo dispositivo de modo que si un controlador falla el segundo lo respalda.

2.2.- DEFINICIONES CONCEPTUALES

OpenFlow Controller[33]: Es una entidad que interactúa con el Switch Openflow usando el protocolo Openflow. En la mayoría de casos un Controlador Openflow es un software el cual controla muchos switches Openflow.

Openflow Switch[34]: Es un conjunto de recursos OpenFlow que pueden ser manejados como una entidad singular, incluyendo el camino de datos y el canal de control.

OpenFlow Channel[35]: Interfaz entre un Switch Openflow y un Controlador Openflow, usada por el controlador para gestionar el switch.

OpenFlow Protocol[36]: Es un protocolo de estándar abierto usado para la comunicación entre los dispositivos que solo poseen plano de datos y el controlador en una SDN.

NETCONF Protocol[37]: Es un protocolo que define un mecanismo simple a través del cual un dispositivo de red puede ser gestionado, la información de los datos de configuración puede ser recuperada y una configuración de datos puede ser actualizada y manipulada.

API: Application Programming Interface, es una interfaz de programación de aplicaciones, son usadas para la comunicación entre el controlador SDN y las aplicaciones en la red (de ser una API dirigida al norte) y para la comunicación entre el controlador SDN y los dispositivos de la capa de infraestructura (de ser una API dirigida al sur).

Red Legacy: Se llama así a las redes tradicionales, entre ellas a las de control distribuido.

Pipeline[38]: Un conjunto de tablas de flujo enlazadas que proveen comparación, reenvío, y modificación de paquetes en un switch OpenFlow.

Action Set[39]: Es un conjunto de acciones que se ejecutarán dentro de un paquete cuando acabe el procesamiento en el Switch OpenFlow.

Metadata[40]: Un registro enmascarable que se utiliza para transportar información adicional de una tabla de flujo de un switch OpenFlow.

2.3 RED DEFINIDA POR SOFTWARE

SDN es una red especificada por *Open Networking Foundation* [41] y su modelo está basada en tres planos y es una red basada en flujo con el controlador centralizado.

2.3.1 PLANOS DE ARQUITECTURA SDN

2.3.1.1 Plano de Datos[42]

Este plano se encarga del almacenamiento en buffer de paquetes, programación de paquetes, modificación de cabeceras y reenvío. La mayoría de los paquetes que llegan al conmutador se manejan sólo en el plano de datos sin ninguna intervención de los otros dos planos; sin embargo, algunos paquetes deben ser procesados en el plano de control, ya sea porque no se tenga información en las tablas del conmutador SDN de cómo manejarlos o simplemente porque pertenecen a un protocolo de control que se debe procesar en el plano de control.

2.3.1.2. Plano de Control[43]

Este plano tiene diversas actividades, pero la principal es mantener actualizada la información contenida en la tabla de flujo, para que el plano de datos pueda manejar de manera independiente el mayor porcentaje de tráfico. Este plano también es responsable de los diferentes procesos de los protocolos de control que afecta la tabla de flujo dependiendo de la configuración y tipo del conmutador.

2.3.1.3. Plano de Gestión[44]

Mediante este plano se realiza la configuración y monitoreo de toda la red, esta capa extrae información de los planos de control y datos de manera apropiada.

Para el nuevo paradigma de las Redes Definidas por Software el plano de control es centralizado y el plano de gestión contiene aplicaciones que se crean en diversos lenguajes de programación dependiendo del controlador a usar, puede ser Python, Java, C++, entre otros.

El control es abstraído de los nodos y se ubica en el controlador, por lo cual los nodos solo quedan con el plano de datos, y la gestión se realiza a través de una aplicación que se comunica mediante una API con el controlador el cual tiene información de la topología de la red.

En la figura 2.2 se ilustra estos planos y las interrelaciones entre ellas.

2.3.1.4 Switch OpenFlow[45]

El *Switch OpenFlow* posee básicamente un canal *OpenFlow* por el cual se comunica con el controlador, además de las tablas de grupo y tablas de flujo, también posee puertos físicos (corresponden a una interfaz de hardware del conmutador), puertos lógicos (no se corresponden directamente con las interfaces del conmutador; son, por ejemplo, agrupación de enlaces como Etherchannel, o puede ser un puerto de túnel o de loopback) y puertos reservados (puertos con acciones de renvío específicas).

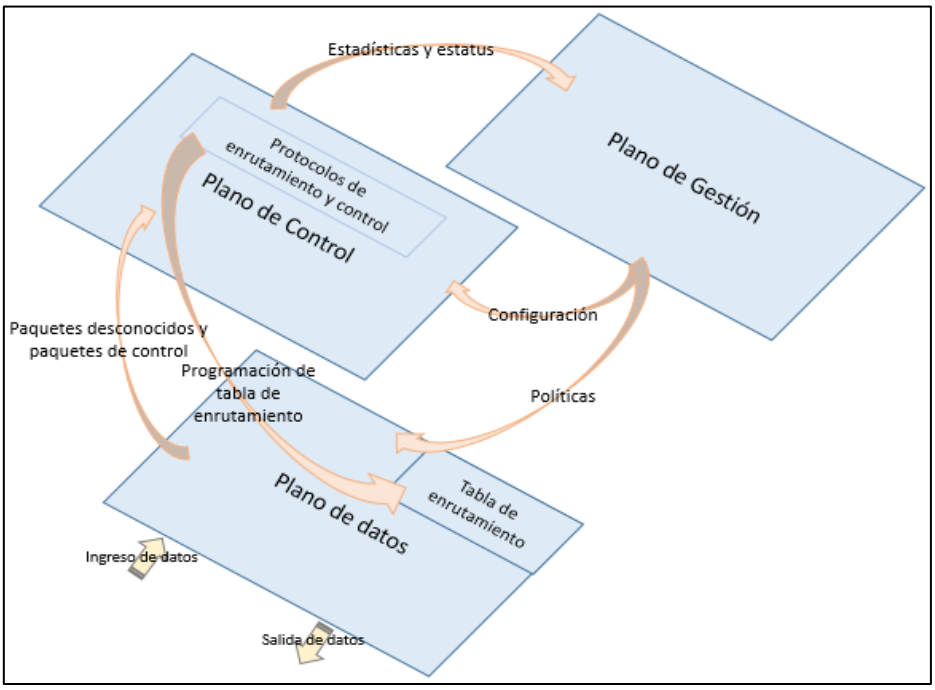


Figura 2.2. Roles de control, gestión y plano de datos. Fuente: Elaboración en base al libro Software Defined Networks A Comprehensive Approach, Second Edition, de Paul Göransson, Chuck Black y Timothy Culver, página 8

2.3.1.5 Tablas de Flujo[46]

Las tablas de flujo contienen entradas de flujo, con información sobre qué es lo que se va a realizar con los paquetes que coincidan con estas entradas

El controlador, es el encargado mediante el canal OpenFlow de actualizar, agregar o borrar las entradas de flujo. Un esquema de una tabla de flujo se ilustra en la figura 2.3.

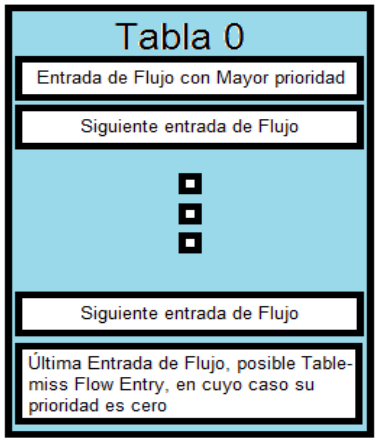


Figura 2.3. Primera Tabla de Flujo, la cual es el número cero. Fuente: Elaboración propia basado en OpenFlow Switch Specification Version1.3.5 (Protocol version 0x04) de la Open Network Fundation

2.3.1.6 Entradas de Flujo[47]

Las entradas de flujo son los elementos que componen una tabla de flujo, estas entradas de flujo contienen un conjunto de campos: de comparación, instrucciones, timeouts, coockie y flags; como se ilustra en la figura 2.4.

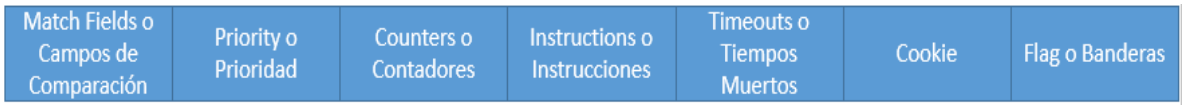


Figura 2.4 Entrada de Flujo. Fuente: Elaboración en base al documento OpenFlow Switch Specification Version1.3.5 (Protocol version 0x04) de la Open Network Fundation

- Match Fields o Campos de Comparación[48]: Son campos que sirven para comparar los paquetes; los paquetes que coincidan serán procesados de acuerdo a las instrucciones de la entrada de flujo, estos campos de comparación contienen: puerto de ingreso, MAC origen, MAC destino, IP origen, IP destino, etiqueta MPLS, ID de Vlan, puerto de origen, puerto de destino, TOS, etc.

- Priority o Prioridad[49]: Este campo indica la prioridad de la entrada de flujo, la tabla de flujo que tenga mayor prioridad será la primera en compararse y la que tenga la prioridad menor será la última; así como las *Access Control Entry* dentro de las *Access List*, la comparación es de la primera a la última, en este caso de la de mayor prioridad a la menor y luego en la primera coincidencia se ejecutará una determinada acción con el paquete.

- Counters o Contadores[50]: Este campo se actualiza cada vez que un paquete coincide con la entrada de flujo en la que se encuentra, por consiguiente, sirve como indicador para saber cuántos paquetes están coincidiendo con una determinada entrada o qué tan usada está siendo una entrada de flujo.

- Instructions o Instrucciones[51]: Estas instrucciones indican qué acciones son las que se va a realizar con el paquete en cuestión, estas instrucciones pueden ser de dos tipos, opcionales u obligatorias. Las opcionales son *Apply–Actions* que se ejecutarán de inmediato, *Meter* dirige al paquete a una medida específica y *Write Metadata* sirve para escribir los valores de metadata en el campo correspondiente. Las obligatorias son *Clear–Actions* sirve para limpiar o borrar todas las instrucciones en el *action set* (o conjunto de acciones) de manera inmediata, *Write–Actions* escribe determinadas acciones en el *action set*;

además, si la acción que se desea escribir en el *action set* ya existe se sobrescribe y *Goto–Table* envía el paquete para que se compare dentro de una tabla en específico.

En la práctica algunas instrucciones son ejecutadas de acuerdo a un orden establecido; por ejemplo, la instrucción *Meter* se ejecuta antes que las *Apply–Actions*, y las *Clear–Actions* se ejecutan antes que las *Write–Actions,* así como la instrucción *Goto–Table* se ejecuta al último.

- Timeouts: Éstos valores indican cuándo vencerá una tabla de flujo y existen dos tipos *Idle timeout* y *Hard timeout*. *Idle timeout* indica que la entrada de flujo expirará, es decir, se borrará del Conmutador OpenFlow luego de no recibir paquetes que coincidan un determinado tiempo con la entrada en cuestión y *Hard timeout* indica el tiempo después del cual una entrada de flujo expirará, sin importar si recibe coincidencias o no.

- Cookie: Este valor no es usado en el procesamiento de paquetes, sin embargo, sirve para identificar cada entrada de flujo y filtrarlas, este valor es elegido por el controlador.

- Flag: Las Flag o banderas en español, modifican el modo de gestión de las entradas de grupo.

2.3.2 TABLAS DE GRUPO[52]

Una tabla de grupo se usa para tratar un grupo de paquetes de la misma manera, así como las tablas de flujo, consiste de entradas, en este caso de entradas de grupo. De éste modo se tiene un método adicional de reenvío, para el cual una entrada de flujo puede apuntar a una tabla de grupo, estas entradas de grupo consisten de los campos: identificador de grupo, tipo de grupo, contadores y cubetas de acción. Las entradas de grupo se identifican por el campo Identificador de Grupo (Group Identifier), como se ilustra en la figura 2.5.

Figura 2.5 Entrada de Grupo. Fuente: Elaboración en base al documento OpenFlow Switch Specification Version1.3.5 (Protocol version 0x04) de la Open Network Fundation

2.3.2.1 Tablas de medición[53]

Una tabla de medición es una tabla que consiste de entradas de medición, las cuales a diferencia de un *action set* no se asignan a un paquete sino a un flujo, lo cual permite implementar diferentes operaciones simples de calidad de servicio como limitación de *rate* (tasa), y puede ser combinado con colas por puerto para implementar estructuras complejas de QoS como *DiffServ* (Servicio Diferenciado). Además, una *medición* mide la tasa de paquetes que tiene asignada, dando así la posibilidad del control de la tasa de esos paquetes, cualquier entrada de flujo puede especificar una medida en su set de instrucciones, la *medida* mide y controla la tasa de agregación de todas las entradas de flujo a la cual apunta.

Pueden existir varios medidores en una misma tabla, pero siendo uno de cada uno. Se pueden usar varios medidores para un grupo de paquetes en tablas de flujo sucesivas por donde pasen. Las entradas de medición se identifican con su Identificador de medición, además poseen otros dos campos, uno para las bandas de medición y el otro para el contador.

Las entradas de medición se componen de tres partes esenciales las cuales son indicadas en la figura 2.6.

Figura 2.6. Entrada de medición. Fuente: Elaboración en base al documento OpenFlow Switch Specification Version1.3.5 (Protocol version 0x04) de la Open Network Foundation.

- Meter Identifier: Este campo es un entero de 32 bits que identifica la *medida.*

- Meter Bands: Un conjunto de *medidas* que no poseen orden donde cada una especifica la tasa de la banda y cómo procesar los paquetes que coincidan.

- Counters: Contadores que se actualizan con cada procesamiento de paquetes de una *medida*.

2.3.2.2 Bandas de medición[54]

Cada medidor puede tener una o más bandas de medida, cada una de ellas especifica la tasa a la cual la banda se aplica y la manera en que los paquetes deben ser procesados, las bandas de medición actúan cuando la tasa de0020paquetes la supera, y se aplica la banda con mayor tasa de banda inferior a la tasa de paquetes de un determinado flujo, como se ilustra en la figura 2.7.

Figura 2.7. Banda de medición en una entrada de medición. Fuente: Elaboración en base al documento OpenFlow Switch Specification Version1.3.5 (Protocol version 0x04) de la Open Network Foundation

Cada banda se identifica por la tasa, pues como se mencionó previamente la banda se aplica siempre y cuando la tasa de paquetes de un flujo alcance y/o sobrepase la tasa asignada en la banda, los campos que posee son los siguientes:

- Band Type: En español tipo de banda, define cómo el paquete va a ser procesado. En la especificación de OpenFlow 1.3 sólo hay dos tipos de banda opcionales, una de ellas es “drop”, el cual sirve para descartar o dropear el paquete, puede ser usado para definir una banda limitante de tráfico y la otra es el “DSCP Remark” el cual incrementa la precedencia del campo DSCP en la cabecera IP del paquete, de este modo puede usarse para definir políticas simples de servicio diferenciado (DiffServ).

- Rate: En español tasa, este es el valor que será usado por el medidor para elegir la banda de medida a utilizar, este valor es el valor mínimo con el cual la banda se aplica.

- Burst: En español ráfaga, define la granularidad de la banda de medida.

- Counters: En español contadores, este campo se actualiza cada vez que un paquete es procesado por una banda de medición.

- Type specific arguments: Este campo sirve para los argumentos adicionales que tienen algunos tipos de bandas.

Como ejemplo, en la Figura 2.8 se muestra la tasa de tráfico en megabits por segundo (eje “Y”) asociado a un flujo “A”, en determinados instantes de tiempo, los cuales están cuantificados en segundos (eje “X”). Además, se aplican dos bandas de medidas asociadas al flujo “A”, la primera con una tasa asignada de 3 Mbps que procesa el paquete remarcando el campo DSCP en su cabecera IP, y la segunda con una tasa asignada de 4 Mbps que tiene asociada el descarte de los paquetes que la sobrepasen, en el ejemplo se muestran las porciones del tráfico (paquetes) a las que se les aplica la banda 1 y la 2.

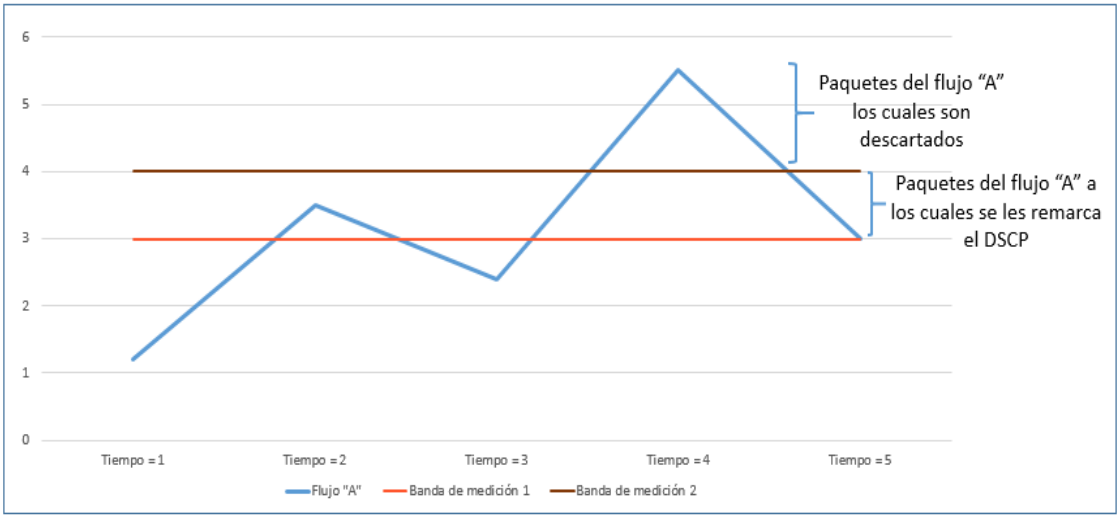


Figura 2.8. Ejemplo de dos bandas de medición asociadas a un flujo. Fuente: Elaboración propia.

2.3.3 PIPELINE[55]

En el esquema de la figura 2.9 se ilustra el procesamiento Pipeline en un Switch OpenFlow. El paquete que ingresa pasa a compararse con la tabla cero, luego al entrar a esta tabla se compara con la primera entrada de flujo, es decir, con la de mayor prioridad, una vez que haga match con cualquier entrada de flujo pasará a compararse con la siguiente tabla, que puede ser la subsecuente o puede ser otra que esté después, esto dependiendo del valor que tenga la instrucción Goto-Table, de no haber hecho match con ninguna entrada, la última

entrada es la *table-miss flow entry*, la cual dependiendo de su configuración indica si se debe descartar el paquete, enviarlo al controlador en forma de Packet-In, o enviarlo a otra tabla, en caso de no haber configurado una *table-miss flow entry* el paquete sin match se descarta. Cada match puede proporcionar dos acciones, una de aplicación inmediata, es decir, se va cambiando la información en los campos del paquete mientras pasa de una tabla a otra y la otra acción es para el *action set*, la cual se va acumulando en el set de acciones el cual es ejecutado al final del pipeline, cabe mencionar que en la versión 1.5 del estándar OpenFlow existe un doble pipeline.

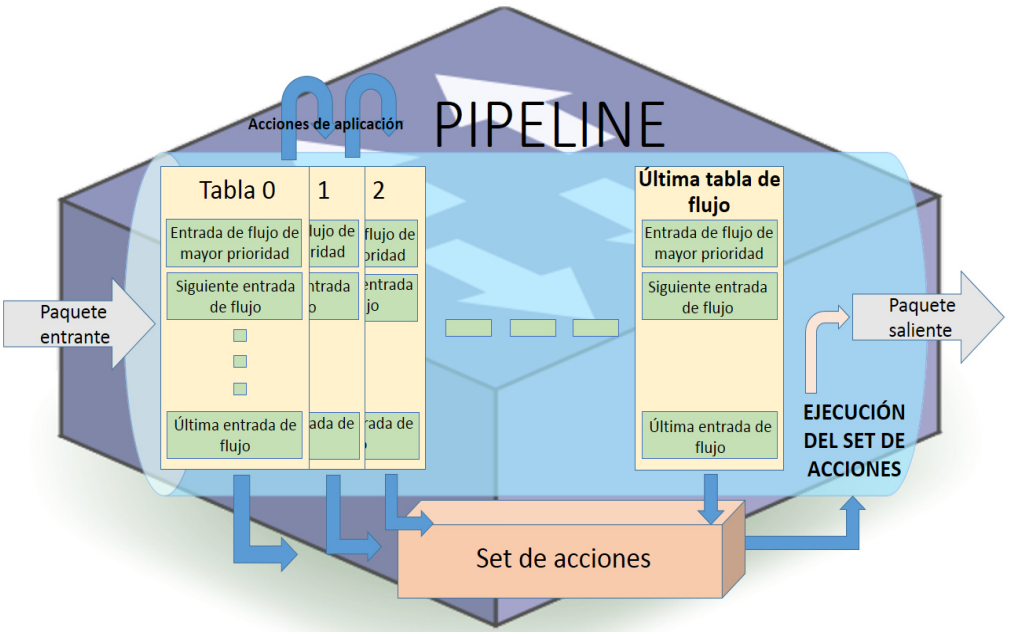


Figura 2.9 Pipeline de un Switch Openflow. Fuente: elaboración propia, basado en la OpenFlow Switch Specification Version1.3.5 (Protocol version 0x04) de la Open Network Foundation

Dentro de los conmutadores OpenFlow existen dos tipos, los que sólo soportan OpenFlow, y los híbridos, en el caso de los conmutadores híbridos el procesamiento OpenFlow es una abstracción, pues el conmutador Openflow se virtualiza sobre el conmutador híbrido.

2.3.4 TIPOS DE MENSAJES OPENFLOW[56]

Hay tres tipos de mensajes Openflow: de controlador a switch Openflow, asíncrono y simétrico.

2.3.4.1 De controlador a conmutador SDN[57]

Son mensajes que como su nombre lo indica, son enviados del controlador al conmutador SDN, estos mensajes son: **Solicitud de características,** para conocer las características del conmutador SDN conectado, **Consulta de parámetros de configuración**, **Modificación de estado**, para agregar, quitar y/o modificar entradas de flujo o de grupo, **Leer estado**, para que el conmutador SDN envíe su estado actual al controlador tanto en características, configuración y capacidades, **Packet-out,** para enviar paquetes que se desea que sean enviados por un determinado puerto del conmutador SDN y para reenviar paquetes recibidos via packet-in, **Barrera**, para asegurar que las dependencias de los mensajes se han conocido, **Solicitud de roles**, usado comúnmente cuando el conmutador SDN se conecta a varios controladores, con este mensaje el controlador configura o consulta el rol del canal OpenFlow, **Configuración asíncrona**, usado para configurar un filtro adicional sobre los mensajes asíncronos que desea recibir sobre el canal OpenFlow o consultar ese filtro.

2.3.4.2 Asíncrono [58]

Son mensajes enviados del Switch OpenFlow al controlador, pero sin que el controlador los haya solicitado previamente y únicamente existen cuatro mensajes de este tipo: **Packet-in**, este mensaje es para darle el control de un paquete al controlador, **Remoción de flujo** para informar al controlador que una entrada de flujo ha sido retirada de su respectiva tabla, **Estatus de puerto** informa al controlador de un cambio sobre un puerto o la configuración de un puerto y **Error** para informar al controlador sobre errores.

2.3.4.3 Simétrico[59]

Estos mensajes se envían en ambas direcciones sin solicitud, y existen tres: **Hello** este mensaje se intercambia entre conmutador SDN y controlador al inicio de la conexión; **Echo** este mensaje puede enviarse de conmutador SDN a controlador o viceversa y es usado para verificar el estado activo del enlace controlador-conmutador SDN, también puede ser usado para medir el ancho de banda o latencia; **Experimentación** este tipo de mensajes está reservado para proporcionar funcionalidades adicionales a los conmutadores SDN.

2.4 CONTROLADORES MULTIPLES[60]

El *hand-over* entre un controlador principal y uno secundario debido a la caída y/o indisponibilidad de gestionar la red por parte del controlador principal es enteramente manejado por los controladores lo que habilita la recuperación rápida ante las fallas además del balanceo de carga.

Todos los conmutadores SDN se conectan a todos los controladores que se le han configurado y este trata de mantener conectividad constante con todos ellos. Los mensajes asíncronos enviados por el conmutador SDN pueden ser enviados a múltiples controladores simplemente duplicando estos mensajes y enviándoselos a todos los controladores conectados mediante canales OpenFlow.

El controlador configurado como “Master”, es decir el principal, es el permitido por el conmutador SDN para realizar cambios en el estado del conmutador SDN y ejecutar comandos controlador-a-conmutador SDN, si el conmutador SDN recibe este tipo de mensajes de parte del o los controladores configurado(s) como “*Slave*” es decir secundario(s) el conmutador SDN responde con un mensaje OFPT_ERROR llevando un campo tipo de OFPET_BAD_REQUEST, sólo se procesan los mensajes que soliciten datos más no los que envíen ordenes de cambio y/o configuración.

Los conmutadores SDN reconocen el rol de los controladores mediante el mensaje OFPT_ROLE_REQUEST y los conmutadores SDN deben recordar el rol de cada controlador conectado.

2.5 CONTROLADORES SDN

Los controladores SDN son los “cerebros de la red” [61], son las entidades que van a tener el control de la red y mediante los cuales se van a dar las ordenes necesarias para que los nodos puedan actualizar sus tablas de flujo y grupo a fin de entregar los paquetes a su destino.

Estos poseen tres tipos de interfaces, las dirigidas al norte, es decir, hacia el usuario (REST API); las dirigidas al sur, es decir, hacia los nodos del plano de datos (OpenFlow, NETCONF, otros); las dirigidas al oeste, es decir, hacia otros controladores que pertenecen a otros dominios de red (diferentes planos de

control SDN por ejemplo otros sistemas autónomos); las dirigidas hacia el este, es decir, hacia el plano de control de dominios de red Legacy[62].

2.5.1 CONTROLADOR OPENDAYLIGHT[63]

La versión utilizada es la Nitrogen SR1 (lanzada el 26 de noviembre del 2018), este controlador es una plataforma de código abierto, basado en Java y brinda soluciones de red listas para instalar (*features*) como por ejemplo servicio AAA, BGP, gestión de datos de internet de las cosas entre otros, soporta los protocolos OpenFlow 1.0 y 1.3. En la figura 2.10 se ilustra sus componentes.

Los *features* de OpenDayLight no vienen pre-instalados ya que algunos pueden tener cierto conflicto y/o interferencia con otros, por lo cual se descargan los que se deseen desde el repositorio: `etc/org.apache.karaf.features.cfg` además el controlador posee un módulo llamado Model-Driven Service Abstraction Layer (MD-SAL) el cual permite a los desarrolladores crear nuevos features, además, se puede considerar que el MD-SAL contiene dos componentes, el primero que es un almacén de datos compartido (a nivel operacional o del estado actual de la red y de configuración o representación del estado deseado de la red) y el segundo que es un bus de mensajes que sirve de plataforma para que los controladores de protocolos y los servicios se comuniquen entre sí [64].

Este controlador es abierto y el que entrega más información tanto de instalación como de uso y desarrollo de aplicaciones.

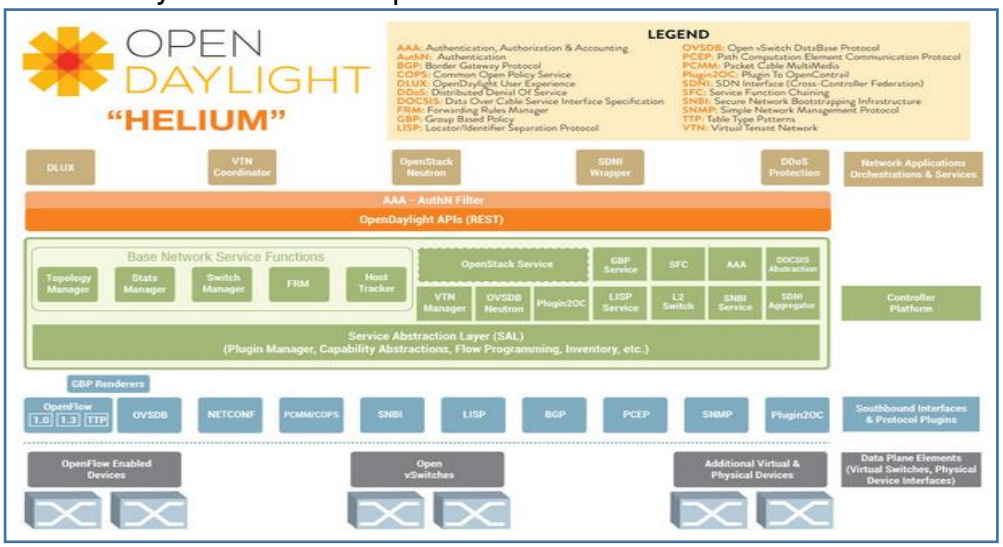


Figura 2.10. Visión de conjunto de las características y compatibilidades del controlador OpenDayLight "Helium". Fuente: https://www.sdxcentral.com/sdn/definitions/sdn-controllers/opendaylight-controller/

Un ejemplo de una topología detectada desde el controlador OpenDayLight se ilustra en la figura 2.11.

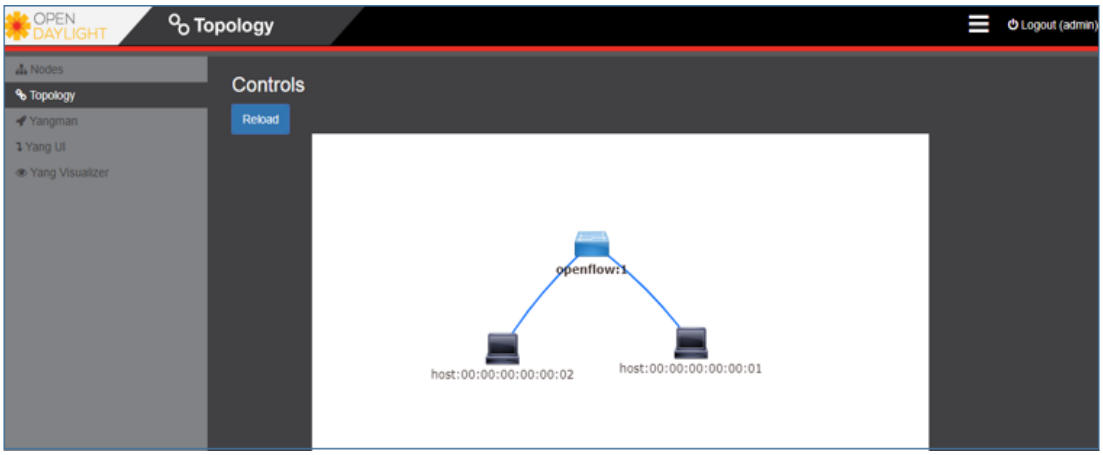


Figura 2.11. Ejemplo de Visualización y gestión de la SDN básica mediante el controlador OpenDayLight.

Existen aplicaciones desarrolladas como el YangUI para el controlador ODL que se pueden usar para obtener información de los nodos, estadísticas de los flujos, estadísticas de las colas, estadísticas de grupo, características del hardware de los Switches OpenFLow, etc.

2.5.1.1 YangUI

YANG es un lenguaje de modelado de datos usado para modificar la configuración y el estado de los datos manipulados por el protocolo de configuración de red (NETCONF), llamadas de procedimiento remoto NETCONF y las notificaciones del mismo protocolo [65].

El controlador OpenDayLight posee un módulo para utilizar el lenguaje YANG con interfaz de usuario el cual es una aplicación desarrollada para facilitar y simplificar el desarrollo de aplicaciones y su prueba[66], a nivel de semántico y en XML respeta el modelo definido en la RFC 6020 [67].

Para instalarlo se debe escribir el siguiente comando en el shell del controlador OpenDayLight: features odl-dluxapps-yangui. Además, el módulo “nodos” permite observar las estadísticas de la red y la información de los puertos para los Switches OpenFlow de la red por lo cual es conveniente también instalarlo con el comando: features odl-dluxapps-nodes.

Esta API (YangUI) se observa en la figura 2.12, con el módulo “nodos” instalado.

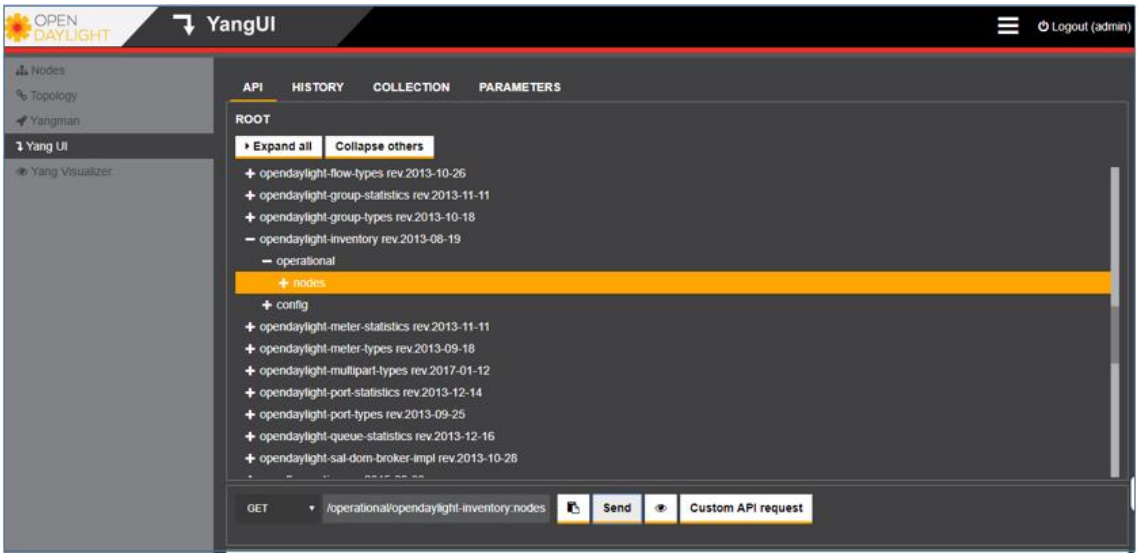


Figura 2.12. Aplicaciones desarrolladas para usar en el controlador SDN OpenDayLight via REST.

Una vez instalado el módulo “nodos” se pueden solicitar la información de los Switches OpenFlow y sus puertos, enviando al controlador una solicitud GET, vía REST (northbound), solicitando los nodos en la red haciendo click en el botón “send” que se observa en la figura 2.12, siendo el resultado de esta petición la figura 2.13, obteniendo información completa de los puertos, direcciones, enlaces, características, estatus, etc.

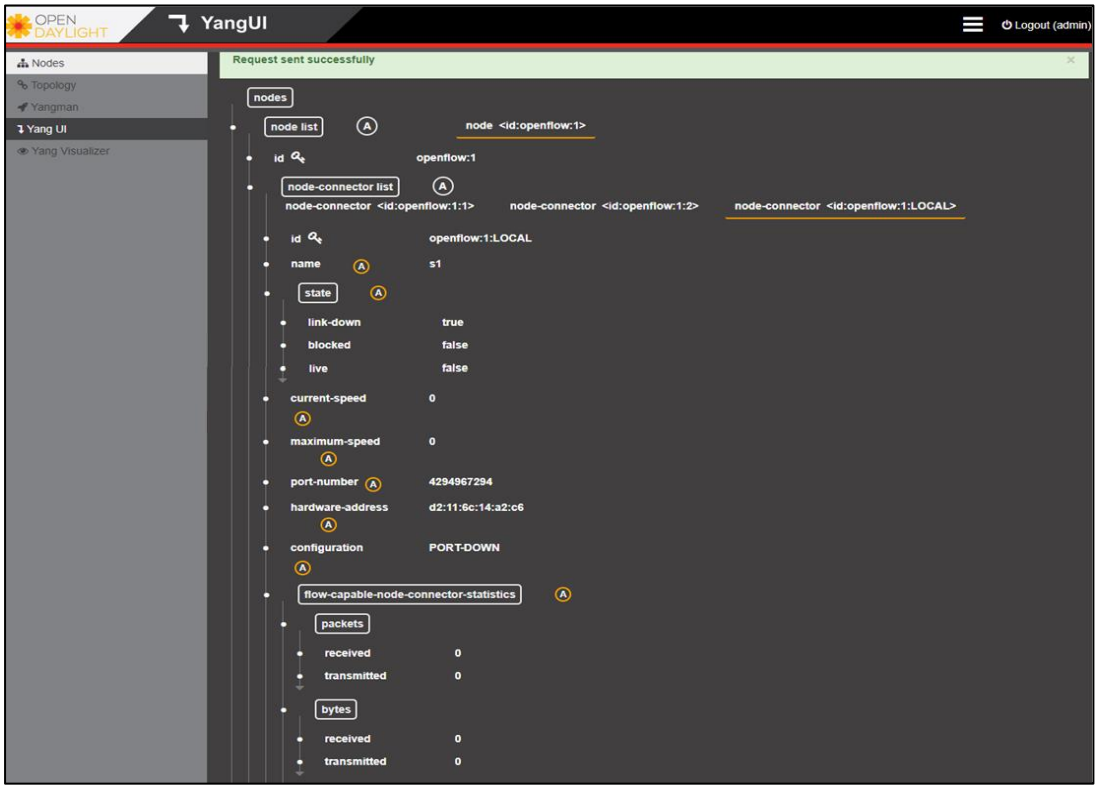


Figura 2.13. Respuesta del controlador SDN OpenDayLight a la solicitud del inventario de nodos.

Por otra parte, también es útil el uso del *OpenFlow Manager* de Cisco el cual permite la visualización de la Red Definida por Software, además de su gestión agregando, quitando o modificando las tablas de flujo y de grupo.

2.5.1.2 OpenFlow Manager de Cisco[68]

Es una aplicación de Cisco que trabaja sobre el controlador OpenDayLight y que sirve para visualizar topologías, programar rutas y reunir estadísticas OpenFlow, como se puede observar en la figura 2.14, la App OpenFlow Manager, se comunica con el controlador a través de la API RestConf.

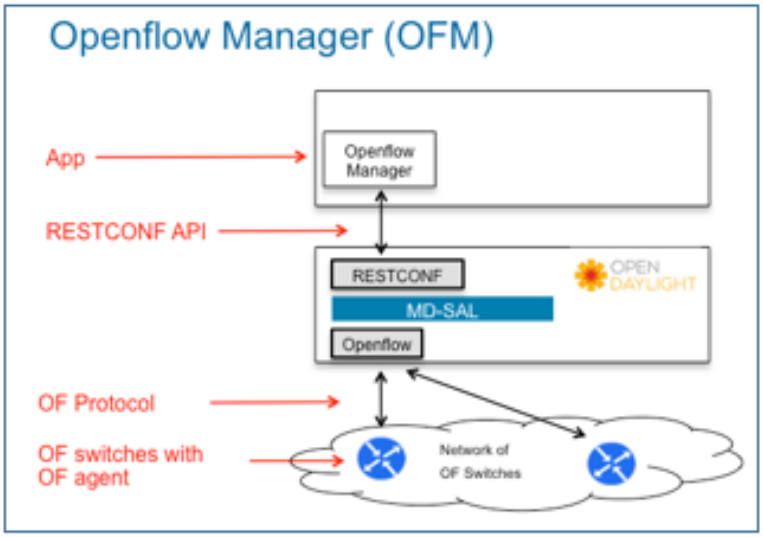


Figura 2.14. Arquitectura del OpenFlow Manager. Fuente: https://github.com/CiscoDevNet/OpenDaylight-Openflow-App

La gestión de la red que entrega esta aplicación se observa en la figura 2.15

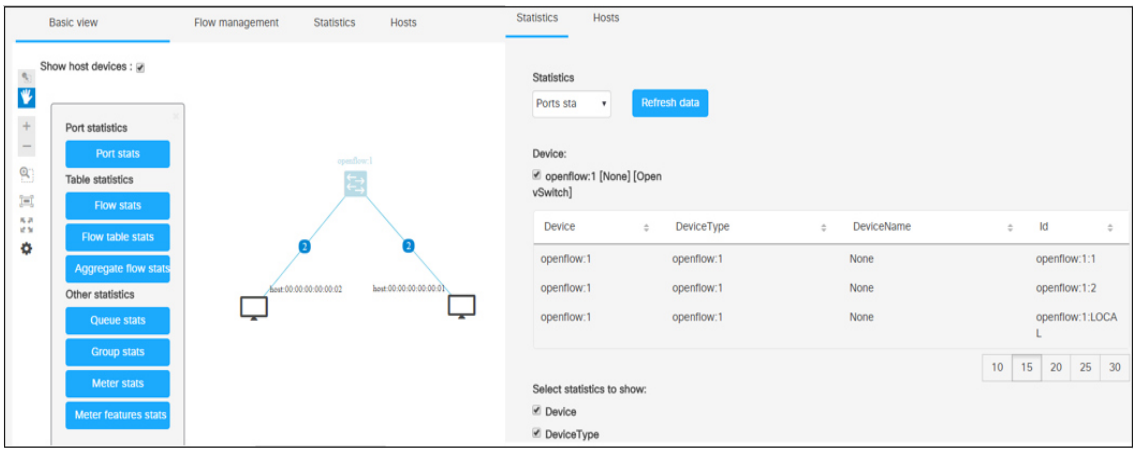


Figura 2.15. Red descubierta por el OpenFlow Manager de Cisco, entregándonos la información del Switch OpenFlow, enlaces, capacidades y opciones de gestión.

2.5.2 CONTROLADOR FLOODLIGHT[69]

La versión utilizada es la 1.1 (penúltima), posee distintas aplicaciones, entre ellas la de ACL (Access-list). Este controlador es constantemente mejorado por una comunidad abierta de desarrolladores, está basado en Java y soporta los protocolos OpenFlow 1.0 y 1.3.

Este controlador al igual que OpenDayLight es licenciado por Apache, y recibe soporte de desarrolladores e ingenieros de Big Switch Networks [70]

Entre las características de este controlador están:

• Ofrecer un sistema de carga de módulos que facilita la ampliación y mejora.
• Facilidad de configuración con mínimas dependencias.
• Capacidad de gestionar redes OpenFlow y no OpenFlow.
• Está diseñado para ser multiproceso y de alto rendimiento, además de soportar la plataforma OpenStack de orquestación en la nube. [71]

En la figura 2.16 se esquematiza el controlador FloodLight en el entorno SDN.

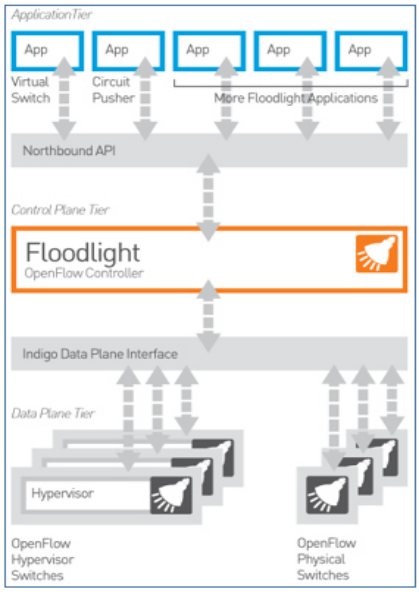


Figura 2.16. Interacción en la SDN del controlador Floodlight. Fuente: http://www.projectfloodlight.org/floodlight/

Este controlador también es abierto pero su desarrollo es menos continuo que la de ODL, la última es de versión es de febrero del 2016.

Detalles de su funcionamiento se explican en el **Anexo B.**

2.5.3 CONTROLADOR ONOS[72]

Este es un controlador Open Source soportado por la comunidad “The Linux Foundation”, el propósito es la conexión entre ISP’s .

Este controlador también está desarrollado en Java y licenciado por Apache y fue diseñado para resolver un problema común entre los controladores que se desarrollaron inicialmente para las SDN, el cuello de botella en el plano de control. Para lo cual ONOS propone una distribución de la carga del plano de control, aunque su limitante sea el tomar la decisión de cómo y cuándo distribuir esta carga, es decir, posee la capacidad, pero falta ganar precisión en el uso de esta. [73]

Es muy usado alrededor del mundo tanto por proveedores de servicios, instituciones de investigación y empresas como Telefónica, NTT, Ciena, Samsung, etc. [74]

La última versión “Nightingale” (nombre que le pusieron en conmemoración al ave del mismo nombre) aún se encuentra en desarrollo.

Entre las características de sus últimas actualizaciones están:

• Soporte IPv6.

• Virtual Private Wire Service (VPWS).

• Ancho de banda garantizado de manera opcional en la interfaz northbound.

• Modelamiento de recursos compartidos, entre otros.

En la figura 2.17 se ilustra los subsistemas del controlador ONOS.

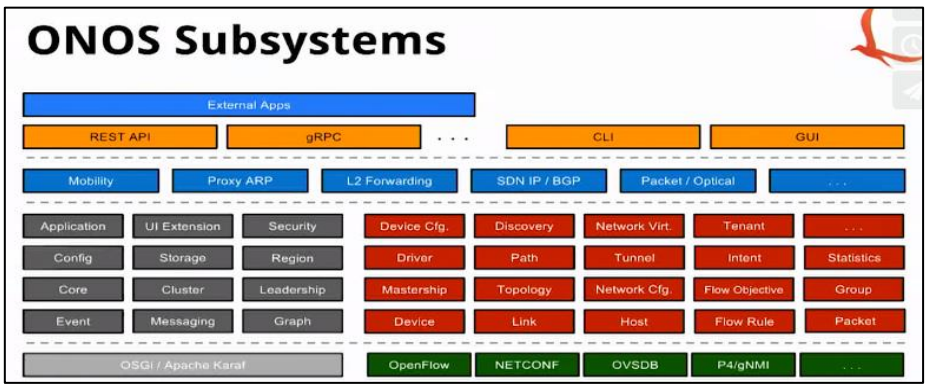


Figura 2.17 Subsistemas del controlador ONOS. Fuente: https://onosproject.org/wp-content/uploads/2018/03/Project-ONOS-Overview-ONF-Vision-Workshop.pdf

2.5.4 Controlador HP Van SDN[75]

Este controlador es desarrollado por Hewlett Packard, está basado en Java, soporta los protocolos OpenFlow 1.0 y 1.3, posee múltiples aplicaciones y soporte por parte de HP (pagando las licencias correspondientes).

Cambió de nombre a Aruba SDN Controller, con las mismas características que tenía, es compatible con los switches de HP (más de 50 modelos), entrega una guía de usuario además de una guía para el desarrollo de aplicaciones aparte de que se tiene el Aruba SDN App Store, también está el SDN Software Development Kit donde se pueden comprar las aplicaciones.

Tiene la capacidad de entregar un modelo que permite compartir la gestión de una red en tres partes, pero usando tres controladores, siendo lógicamente uno sólo, es decir, compartiendo una vista común de la red. En caso falle uno de estos controladores, se genera una respuesta rápida por parte del *cluster* para proveer la continuidad en la operación de la red [76]. En la figura 2.18 se ilustra la arquitectura del controlador HP VAN.

Tiene un módulo de topología de servicio optimizado para poder buscar hosts, Switches OpenFlow o usuarios en el GUI [77]. A diferencia de otros controladores propietarios como Cisco, el de HP es más accesible ya que se encuentran más guías de uso en la web.

También ofrece un módulo llamado "*Follow Flow*", para decidir la ruta que deben seguir los paquetes que van desde un determinado origen a un respectivo destino, la otra opción es usar el "*Shortest Path*" lo que implica tomar la ruta más corta de un punto a otro, además de su apartado para el monitoreo llamado "*Alerts*". Detalles de su funcionamiento se explican en el **Anexo A**.

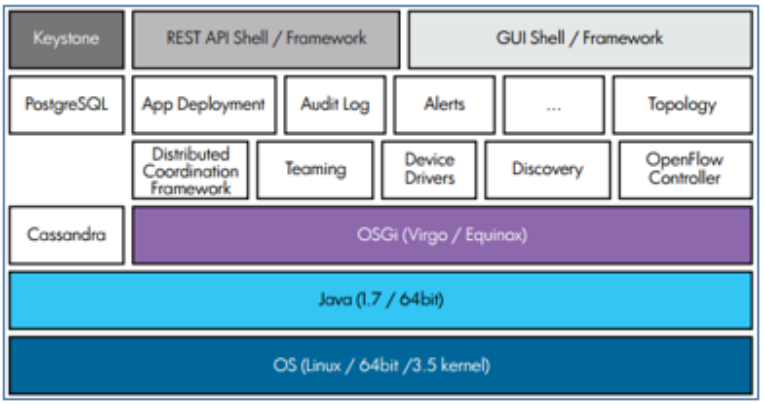


Figura 2.18. Arquitectura del controlador HP VAN. Fuente: HPE VAN SDN Controller 2.6 Administrator Guide

2.6 ANÁLISIS BÁSICO DEL FUNCIONAMIENTO DE SDN

Para el análisis del funcionamiento de SDN se partirá de la topología básica ilustrada en la figura 2.19, utilizando como controlador a OpenDayLight con un OvSwitch (OpenFlow Switch 1) y dos hosts (Host 1 y 2). Se analiza los mensajes capturados por la herramienta Wireshark entre el controlador y el OpenFlow Switch 1.

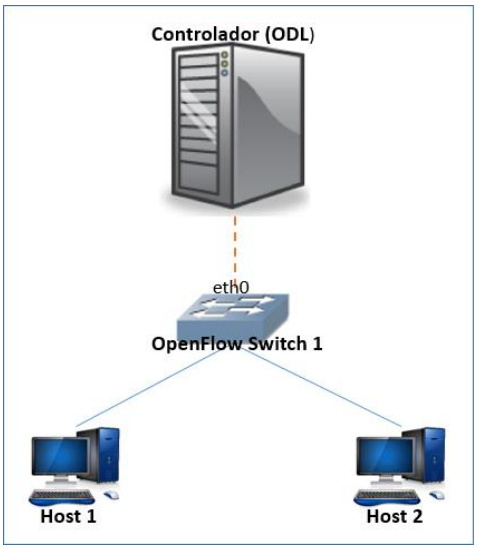


Figura 2.19 Red básica para caso de estudio. Fuente: elaboración propia.

2.6.1 HABILITACIÓN DE TOPOLOGÍA SDN

Esta es una topología que vienen construidas en el Mininet y para iniciarla solamente hay que introducir el comando: `sudo mn --controller=remote,ip=192.168.1.40 --ipbase=192.168.1.128/28 --switch=ovsk,protocols=OpenFlow13 –mac`. Este comando crea la red lista para trabajar con un controlador externo[78], sin embargo, el controlador aún no descubrirá la topología hasta que se haga un ping entre un host y el otro para lo cual se ejecuta el comando `h1 ping h2`, como se ilustra en la figura 2.20.

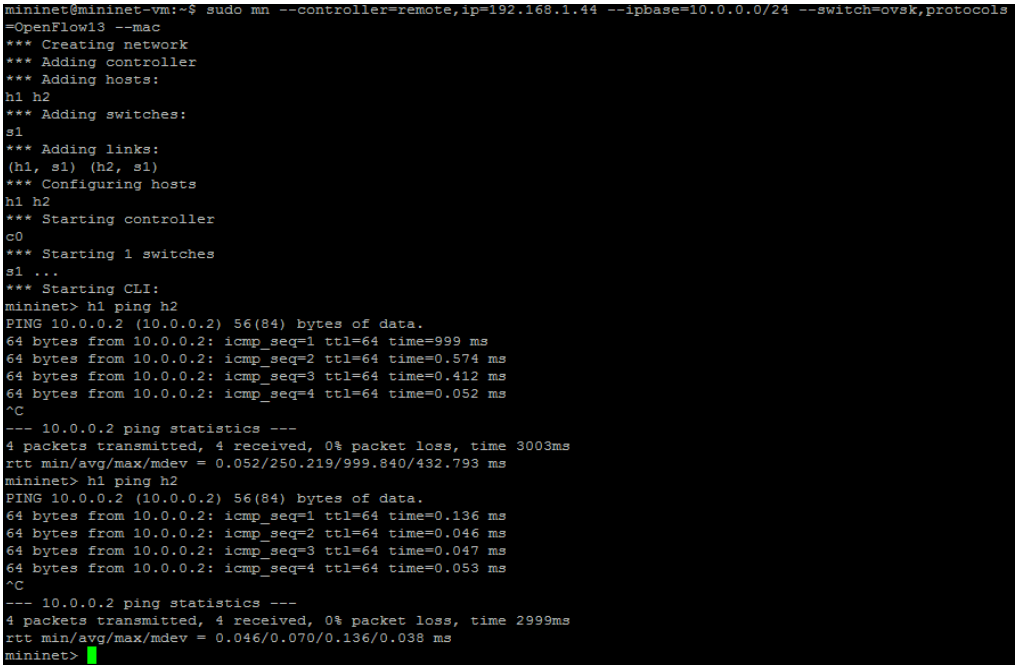


Figura 2.20. Comandos ejecutados, en el servidor que contiene al simulador Mininet.

En la figura 2.20 se puede observar que luego de la ejecución del comando para la creación de red, esta se realiza, y que al hacer el ping entre el host 1 y el host 2, el RTT del primer mensaje ICMP es mucho mayor a los demás, esto se da porque en la red aún no hay tablas, para que haya tablas el controlador debe de enviarlas al OpenFlow Switch 1. Al realizar el ping, el host 1 envía un mensaje ICMP hacia el host 2, debido a que el OpenFlow Switch 1 aún no tiene una tabla que le informe qué hacer con el paquete que recibe por parte del host 1, envía al controlador una captura de una porción del paquete, luego el controlador le envía la tabla de flujo correspondiente informándole por qué puerto debe reenviar el paquete de manera que sea entregado al host 2. Este proceso se analiza en el punto 2.6.2.

2.6.2 ANÁLISIS DE LOS MENSAJES OPENFLOW

Para el caso de la topología de la figura 2.14, el Switch OpenFlow (OpenFlow Switch 1) y los hosts (Host 1 y 2) se encuentran en una máquina virtual con IP: 192.168.1.46 y el controlador OpenDayLight en una máquina virtual con IP: 192.168.1.41 y se realiza la captura en la interfaz eth0 que conecta el switch OpenFlow al controlador.

• **Cabecera del mensaje OpenFlow[79]:** La cabecera de los mensajes bajo el protocolo OpenFlow contiene los siguientes campos que se observan en la figura 2.21.

- Versión: Este campo de 8 bits nos indica la versión del protocolo OpenFlow, para la presente tesis hemos usado la versión 1.3

- Tipo: Este campo de 8 bits nos indica el tipo de mensaje OpenFlow que se está mandando, puede ser: Hello, solicitud de características, Packet-In, etc.

- Longitud: Este campo de 16 bits nos indica la longitud en bytes del contenido del protocolo OpenFlow del paquete.

- Xid (Transaction ID): Este campo de 32 bits es un identificador de cada mensaje OpenFlow, para una posible respuesta, de manera similar al campo identificador en la cabecera IP de los paquetes IPv4.

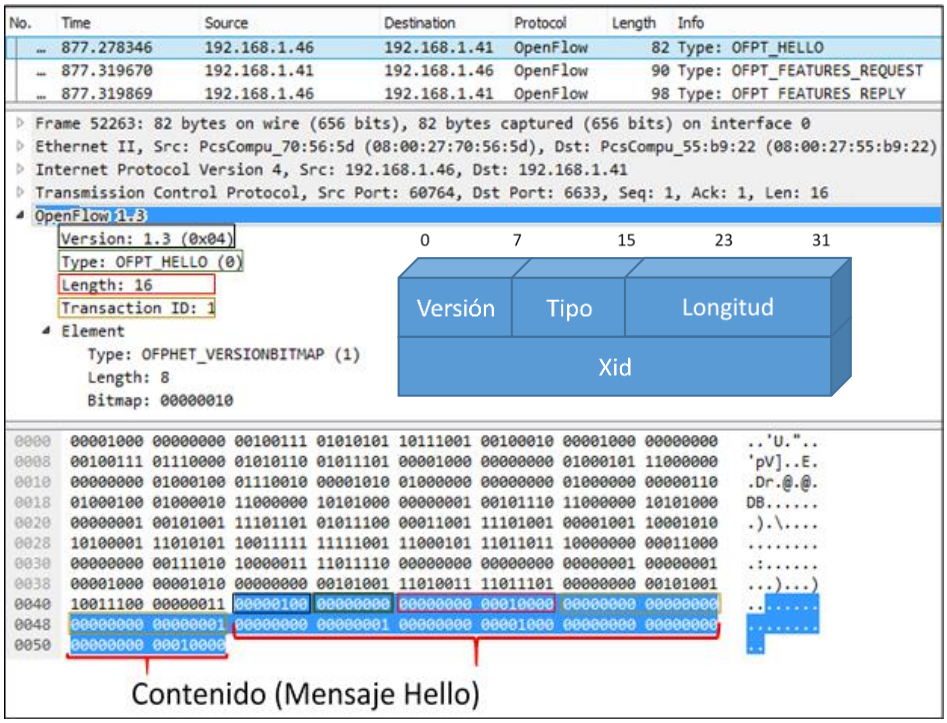


Figura 2.21. Cabecera del primer mensaje OpenFlow.

En la figura 2.21 se observa la composición del mensaje *Open Flow*, teniendo en primer lugar la cabecera compuesta por los campos mencionados y en el payload el mensaje *Hello*, este es un mensaje simétrico que se intercambia sin previa solicitud[80], en este caso enviado por el Switch OpenFlow (OpenFlow Switch 1) para establecer la conexión con el controlador OpenFlow en el puerto 6633.

Continuando el análisis, en la figura 2.22, se observa a nivel de trama cómo se identifica al controlador (PcsCompu_55:b9:22) y al OpenFlow Switch 1 (PcsCompu_70:56:5d), siendo la comunicación dentro de una sesión TCP establecida con puerto de destino: 6633.

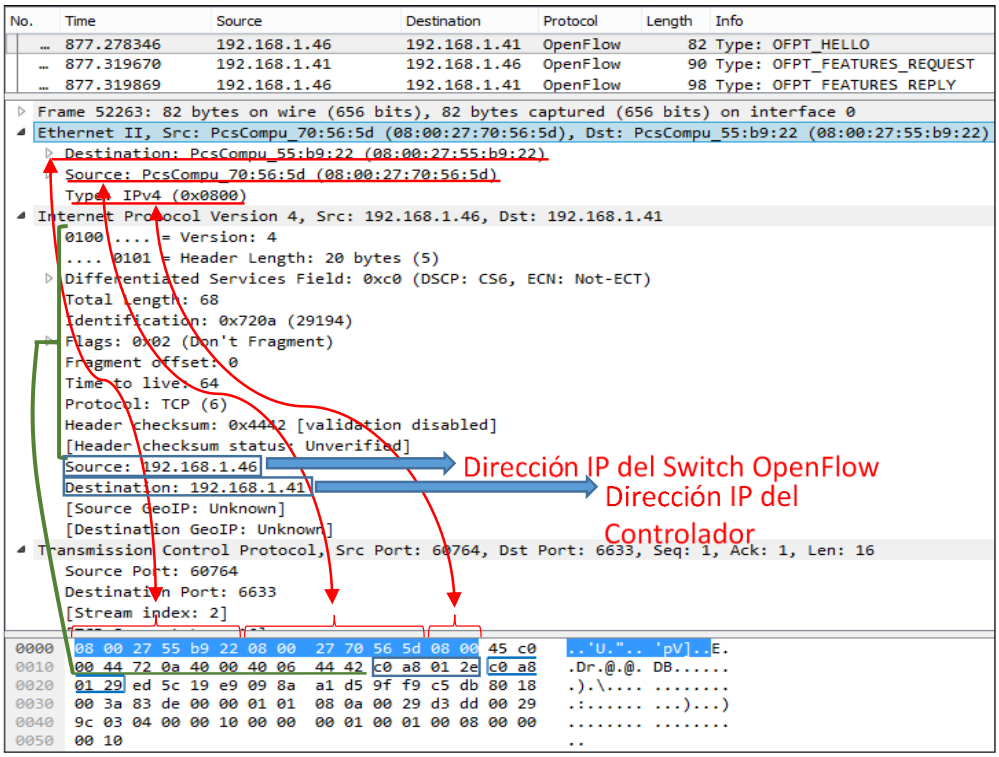


Figura 2.22. Primer paquete con contenido OpenFlow enviado en la arquitectura SDN.

En la figura 2.23 se observa el segundo mensaje enviado el cual parte del controlador hacia el *Switch OpenFlow* solicitando sus características. Contiene dos mensajes: *Hello* (que vendría a ser la respuesta por parte del controlador para establecer la adyacencia) y *solicitud de características* con id de transacción: "2".

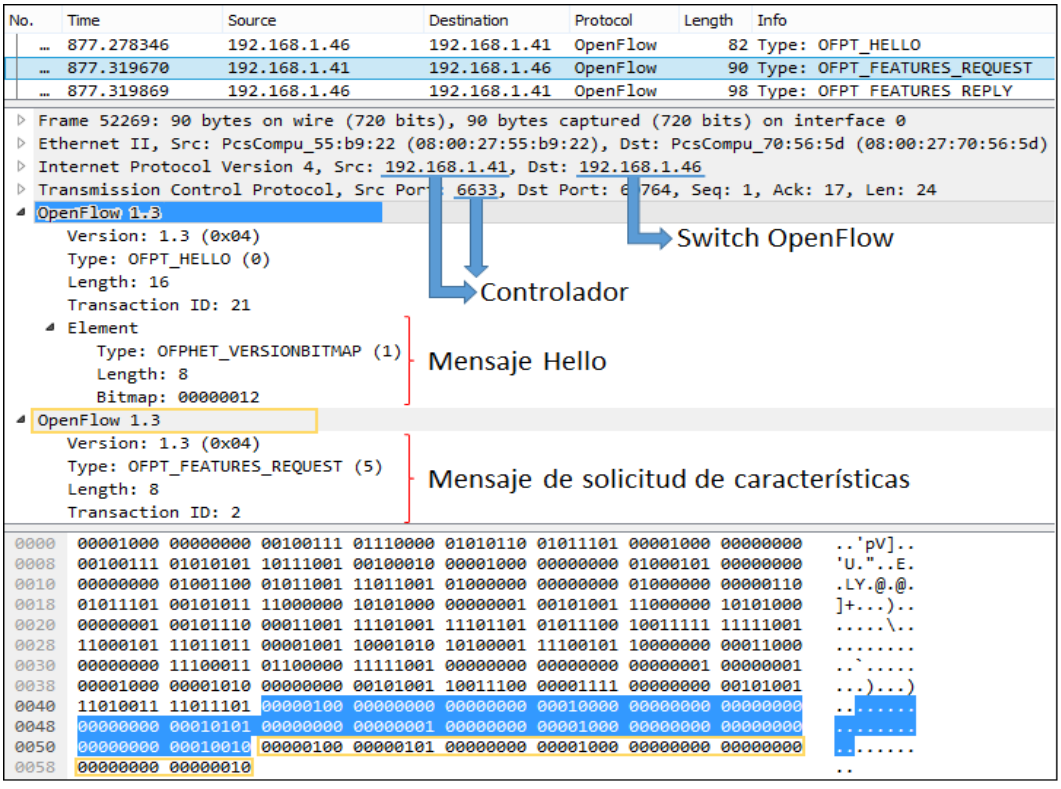


Figura 2.23. Segundo paquete con contenido OpenFlow enviado en la arquitectura SDN.

En la figura 2.24 se observa la respuesta del switch Openflow hacia el controlador con un id de transacción “2”, este id identifica al mensaje como contestación al mensaje previo que tiene el mismo id de transacción.

Dentro del grupo de capacidades del Switch OpenFlow tenemos estadísticas de flujo, tablas, puertos, grupos, capacidad de reensamblar fragmentos IP, estadísticas de colas y el último campo de bloqueo de puertos, nos informa si hay un protocolo externo a OpenFlow como el *Spanning Tree Protocol* para evitar bucles en la topología, en nuestro caso el valor es cero ya que no hay tal protocolo.

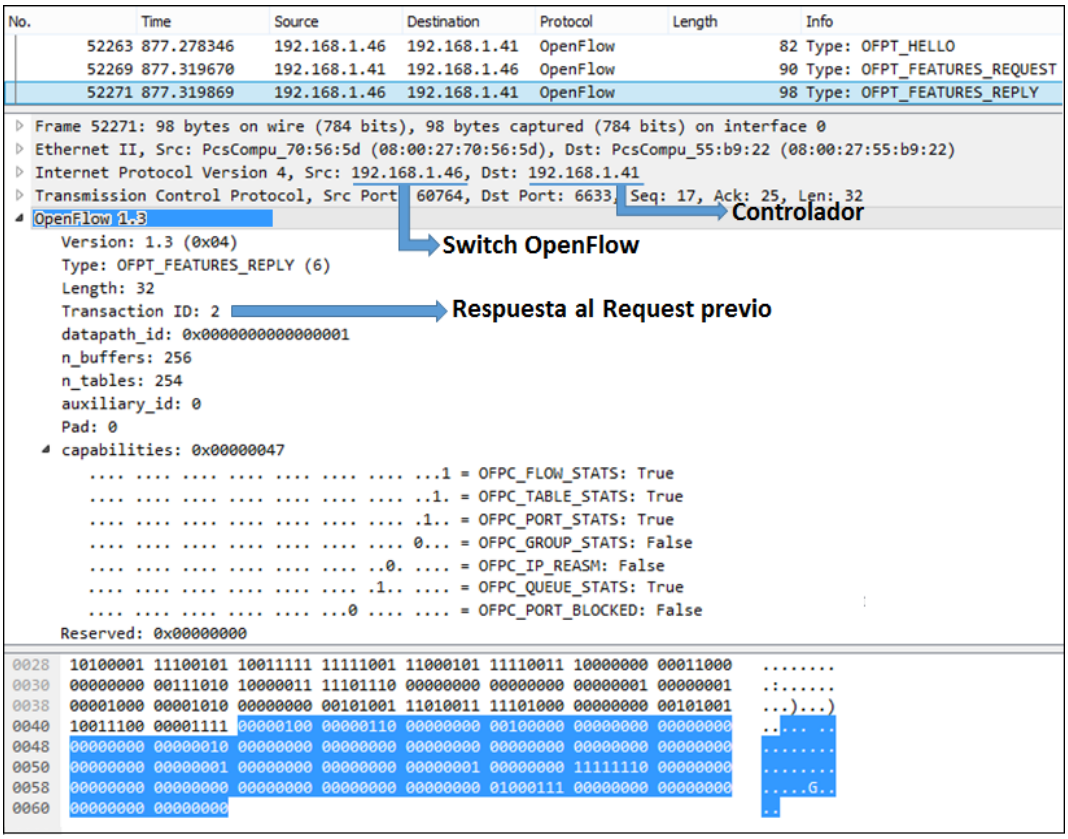


Figura 2.24. Tercer paquete con contenido OpenFlow enviado en la arquitectura SDN.

En el mensaje observamos el campo ***datapath_id*** el cual hace referencia al pipeline por el cual ha pasado ese mensaje (los primeros 48 bits son de la MAC del OpenFlow Switch 1 los siguientes 16 representa el Vlan ID), lo que implica que cada Switch OpenFLow manda un mensaje con un valor distinto con la posibilidad de que un Switch OpenFLow mande dos mensajes con diferentes

números de *datapath_id*, esto se daría si tiene dos pipelines, es decir, diferentes Switches OpenFlow virtualizados en el mismo switch físico (particionado lógicamente los recursos de hardware), el campo ***n_buffers*** nos indica la cantidad de paquetes que el Switch OpenFLow puede poner en cola para procesarlos como *Packet-In*, el campo ***n_tables*** nos indica la cantidad de tablas que puede soportar el Switch OpenFlow, el campo ***auxiliary_id*** nos indica el tipo de conexión *OpenFlow* entre este y el controlador, puede ser una conexión principal o auxiliar, como en nuestro caso es una conexión principal, el valor de este campo es cero, el campo *PAD* es usado para mantener el alineamiento de los bytes cuando es necesario[81].

En la figura 2.25 se observa que la comunicación continúa con los mensajes: *Barrier Request y Barrier Reply* utilizados como un punto de referencia para la sincronización[82], *Multipart Request* solicitud de estadísticas el cual es un mensaje enviado del controlador al Switch *OpenFLow* y puede solicitarse información diversa y el *Multipart Response* en nuestro caso sobre la descripción del *Switch OpenFlow*, es decir, el nombre, fabricante, etc.

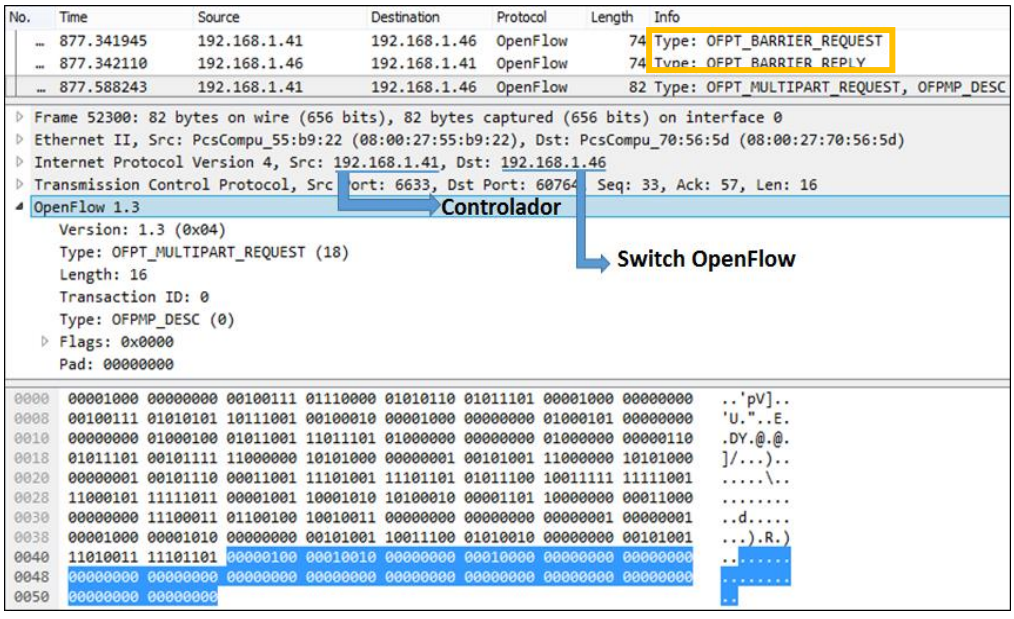


Figura 2.25. Paquete de solicitud de estadísticas de descripción enviado por el controlador.

En la figura 2.26 se observa la respuesta del OpenFlow Switch 1 al mensaje de solicitud de características de descripción enviado previamente por el

controlador. Dentro de la información enviada en el Multipart Reply está el fabricante del OpenFlow Switch 1, la descripción del hardware, el nro de serie, etc.

En la figura 2.27 se tiene el mensaje PORT_STATUS enviado por parte del OpenFlow Switch 1 como reconocimiento del host 1 conectado.

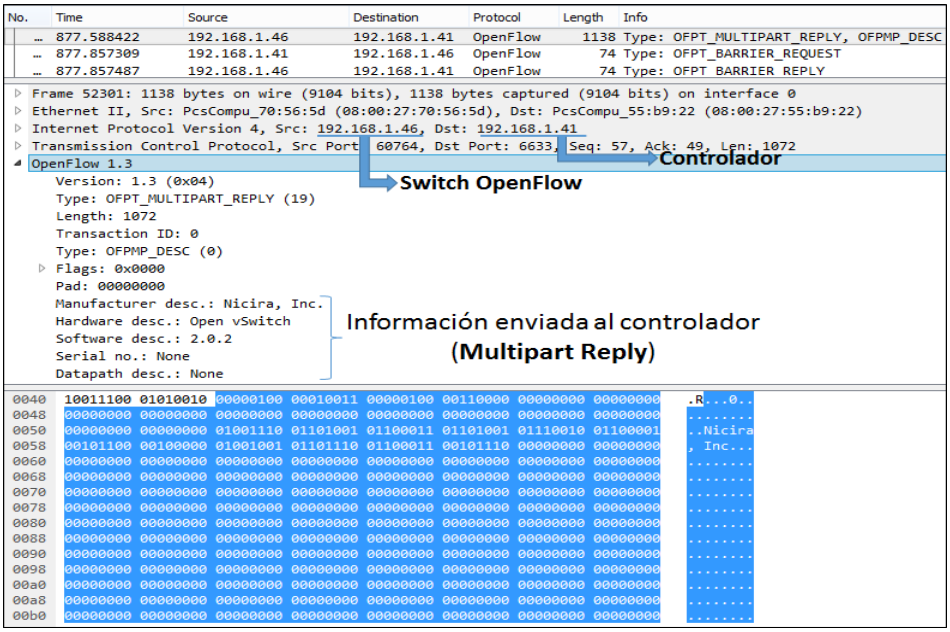


Figura 2.26. Paquete de respuesta de estadísticas de descripción enviado por el Switch OpenFlow.

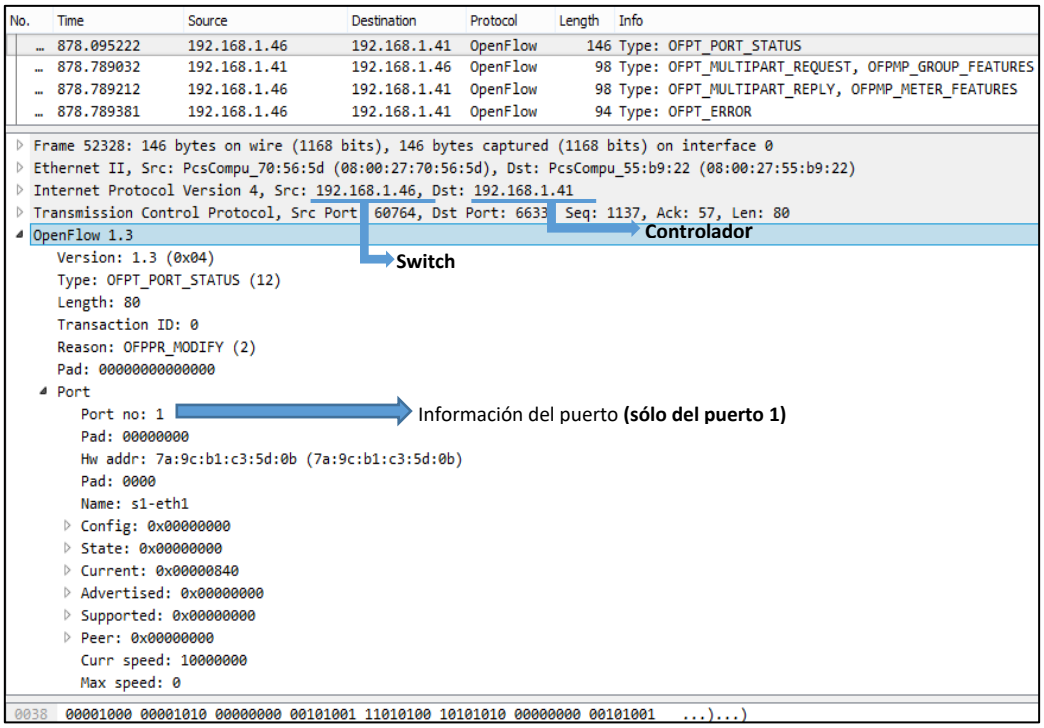


Figura 2.27. Paquete asíncrono PORT_STATUS enviado por el OpenFlow Switch 1 al controlador

Cuando el OpenFlow Switch 1 detecta un cambio en un puerto (por ejemplo, la conexión de un PC) envía el mensaje Port_Status, en el contenido tiene la dirección MAC del puerto en mención, su número de puerto y su nombre[83].

Luego de que se hace un ping desde el host 1 hacia el host 2, se genera un **Packet-In** que el OpenFlow Switch 1 envía al controlador el cual se observa en la figura 2.28.

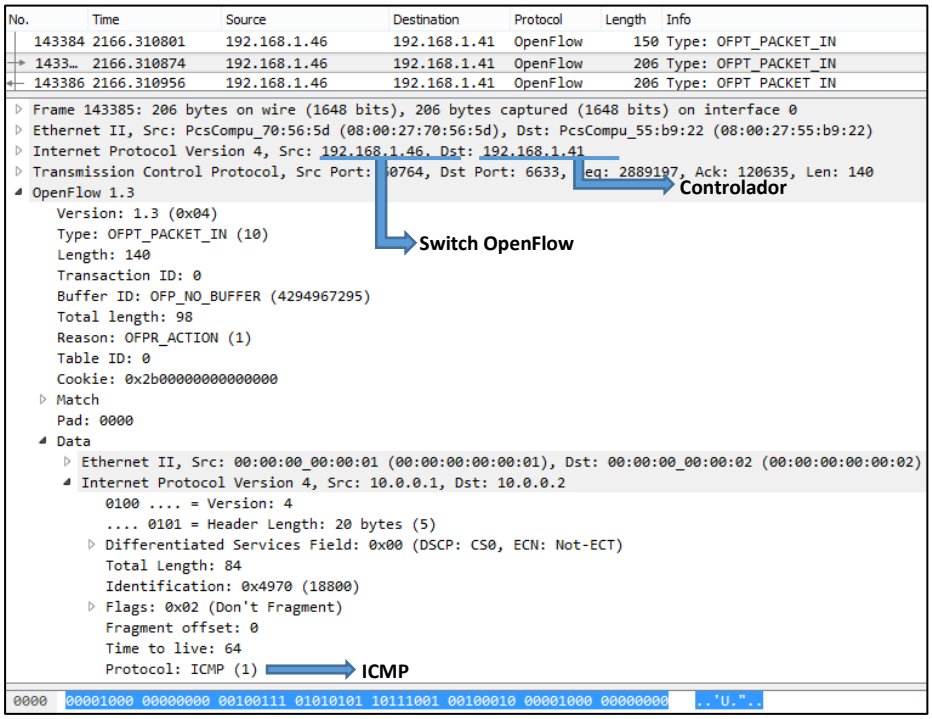


Figura 2.28. Paquete OFPT_PACKET_IN enviado por el OpenFlow Switch.

En el mensaje podemos observar que es resultado del OpenFlow Switch al recibir un ICMP desde el Host 1, producto de ejecutar el comando ping en el terminal del Host 1, sin embargo, al no tener tablas de flujo captura parte de ese paquete y lo envía como Packet_In al controlador a la espera de instrucciones.

El controlador responde con un mensaje **Packet_Out** el cual se observa en la figura 2.29.

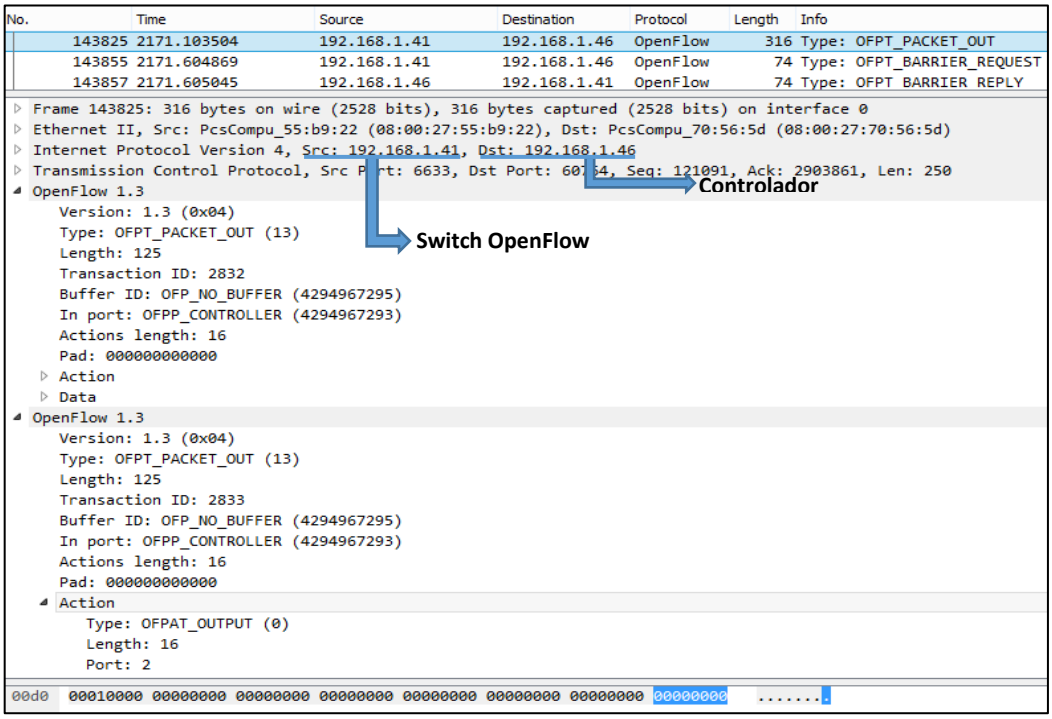


Figura 2.29. Paquete OFPT_PACKET_OUT enviado por el OpenFlow Switch.

Este mensaje Packet_Out contiene la acción enviada al OpenFlow Switch 1 que consiste, a manera de ejemplo, enviar el mensaje por el puerto 2, al cual está conectado el Host 2, también son enviadas instrucciones de modificación de tablas de flujo como se observa en la figura 2.30.

Este mensaje OFPT_FLOW_MOD (figura 2.30) instruye al OpenFlow Switch 1 de que modifique la tabla de flujo con Id: “0”, es decir, la primera tabla, además los tiempos *Idle* y *Har*d tienen el valor cero lo que indica que esta tabla de flujo debe ser permanente y dentro de la instrucción[84], vemos que se aplica a las Apply Actions, con dos acciones, la primera que es reenviar el paquete por el puerto nro 2 del OpenFlow Switch 1 y la segunda que es enviarlo al puerto del controlador, por consiguiente los demás paquetes que entren por el puerto nro 1 seguirán esta entrada de flujo.

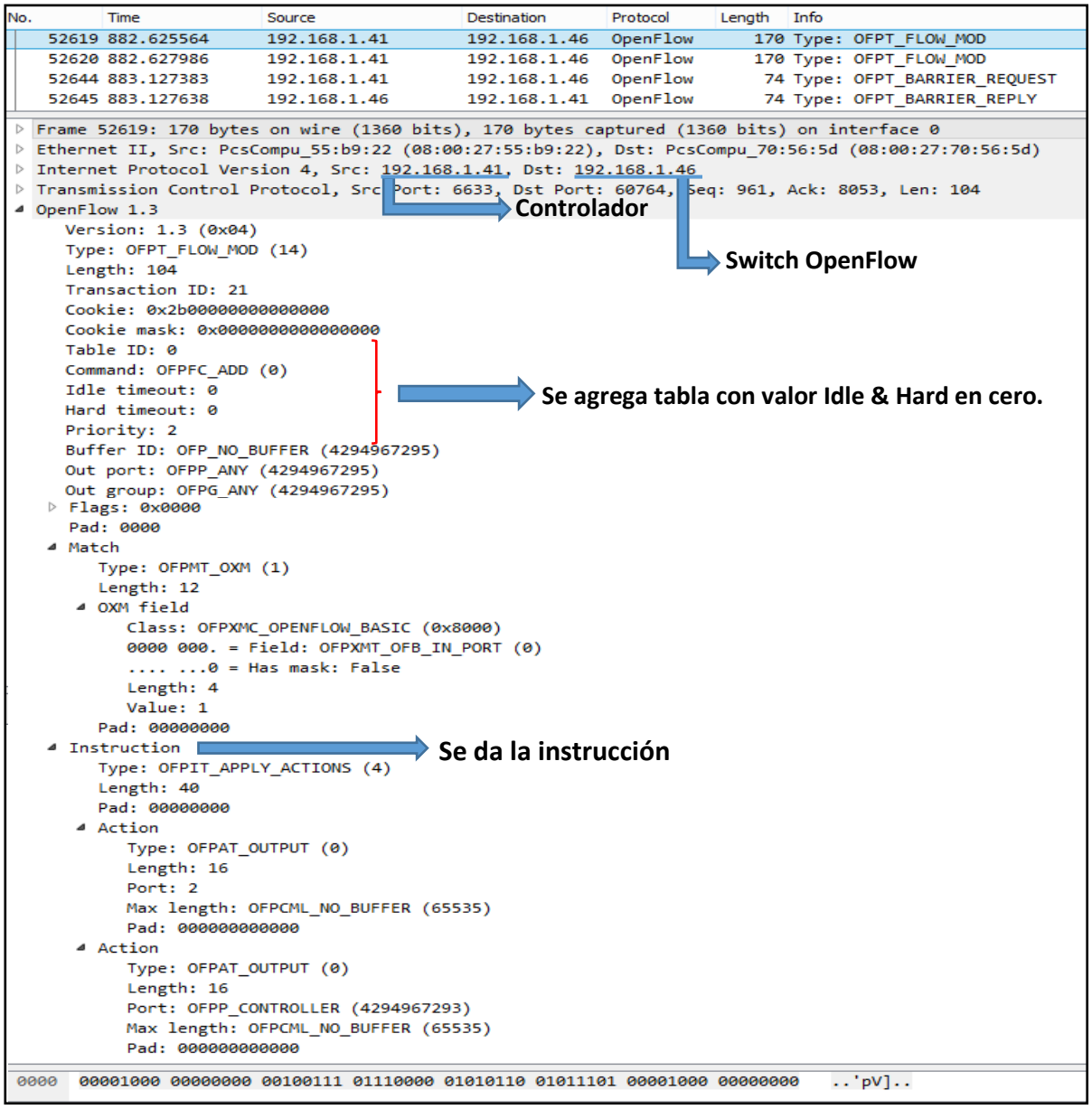


Figura 2.30. Paquete OFPT_FLOW_MOD enviado por el controlador.

2.6.3 REST API

Las actualizaciones de las tablas de flujo o de grupo se realizan en los controladores, para ello tienen un apartado en el cual se hacen solicitudes en formato JSON (*JavaScript Object Notation*), estas son estructuras de peticiones con un formato determinado, sin embargo, también se puede usar el entorno de desarrollo de API's Postman, en caso de hacerlo directamente, uno de los más desarrollados es el del OpenDayLight a través del Yangman, el cual es una

aplicación que ofrece formas de interfaz de usuario dinámicamente generadas, y su representación nativa en JSON basada en REST API's.

Las interfaces de programación de aplicaciones basadas en la arquitectura REST (Transferencia de estado representacional) debe de respetar los siguientes principios [85]:

a) La primera restricción a agregarse a este sistema es la de respetar la arquitectura Cliente-Servidor.
b) Debe ser Sin-Estado, lo que implica que cada petición de usuario debe poseer la información completa y necesaria y no depender de ninguna otra solicitud previa.
c) Debe poseer Caché para determinada información de modo que si la información es reconocida como candidata para ser almacenada en el Caché los datos de esa consulta pueden ser reutilizados en una próxima consulta.
d) Debe poseer una Interfaz Uniforme, lo que implica que las interfaces entre componentes deben ser uniformes de modo que aplicando el principio de ingeniería de software de generalidad a la interfaz de componentes la arquitectura general del sistema se simplifique.
e) Debe ser un Sistema con Capas, esto implica que la implementación de este sistema debe constar de capas, de modo que esta arquitectura esté compuesta por capas jerárquicas restringiendo el comportamiento de componentes los cuales no puedan "ver" más allá de la capa inmediata con la que interactúan.
f) Debe poseer el estilo de Código-Sobre-Demanda, esta es una restricción opcional que implica que se permita que la funcionalidad del cliente se extienda descargando y ejecutando código en forma de scripts o applets.

La arquitectura REST API es asumida en la implementación de los controladores SDN, permitiendo que la información que un controlador SDN obtiene de los switches OpenFlow conectados, sean leídas y mostradas adecuadamente por las aplicaciones ubicadas en la capa de aplicación de la arquitectura SDN. De esta manera la arquitectura REST API corresponde a la interfaz northbound en SDN.

CAPÍTULO III

PROPUESTA DE LABORATORIO SDN

3.1 INTRODUCCIÓN

Existen cuatro tipos de redes SDN. En primer lugar, está la Red Canónica, en esta red todos los nodos poseen control centralizado, es decir, es una red sin nodos legacy y/o mixtos. En segundo lugar, está la Red Compilada, esta es una red donde todos los nodos poseen control distribuido y centralizado, y se encuentran bajo el controlador. En tercer lugar, están las redes híbridas, donde todos los nodos poseen capacidad de control distribuido y centralizado, pero no todos se encuentran bajo el controlador. Por último, tenemos a la Red Sobrepuesta (Overlay), donde algunos nodos poseen sólo control distribuido y otros sólo control centralizado.

En líneas generales la propuesta de laboratorio es una topología estrella con niveles: *core*, distribución y conexión, encontrándose el core y distribución juntos ya que los nodos que gestionan toda la información (nodos core) serán los mismos que recolectarán el tráfico (nodos distribución) de los nodos que se conecten a los equipos de usuario (nodos de conexión). En la figura 3.1 se muestra un esquema de la propuesta.

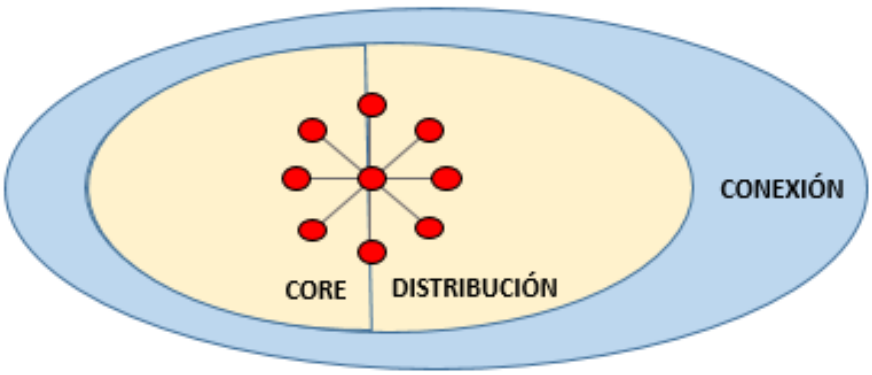


Figura 3.1. Esquema de los niveles de jerarquía en la Propuesta de Laboratorio SDN.

3.2 LABORATORIO SDN BASADA EN OPEN SOURCE

Dentro de los escenarios SDN existen dos vertientes, por un lado, están los escenarios Open Source promovidos por comunidades desarrolladores de software y por otro lado los provistos por los fabricantes de estas tecnologías que dan como origen soluciones propietarias. Estas soluciones propietarias (por citar, las ofrecidas por Cisco o Juniper) tienen características muy útiles, por ejemplo, actualizaciones y mantenimiento inmediato de los programas del controlador; pero al no ser Open Source el administrador de la red está limitado a acceder y modificar el código del controlador para dar soluciones propias en la optimización de la red o proponer nuevas soluciones innovadoras. Otro aspecto limitante de una solución propietaria es que se debe pagar por las actualizaciones de las licencias del controlador cuando se incrementa la cantidad de nodos, y/o obtener nuevas características en la red.

Para los investigadores y del personal en proceso de capacitación, mantener la flexibilidad y la capacidad de estar en contacto continuamente con el código fuente del controlador siempre es primordial. En cambio, para un operador de computación en la nube estar en contacto con el código es mucho menos importante que tener una organización de soporte confiable a la cual llamar ante una falla en la red [86].

Por lo expuesto, en la presente tesis se propone que el laboratorio SDN para la Facultad de Ingeniería Electrónica y Eléctrica de la UNMSM debe ser del tipo Open Source. De esta manera los alumnos de pregrado y postgrado, así como los profesores investigadores de esta facultad, podrán realizar sus acciones de capacitación e investigación sin ningún tipo de limitaciones debido a que se tendrán acceso al código del controlador y switches SDN.

3.2.1 LENGUAJE DE PROGRAMACIÓN P4

Su nombre quiere decir "*Programming Protocol-Independent Packet Processors*"; es decir, protocolo de programación independiente de los procesadores de paquetes.

El Consorcio de Lenguaje P4 lo define como un "*lenguaje declarativo para expresar cómo los paquetes son procesados por el conducto de un elemento de red de reenvío*", ya sea este un switch, router, NIC, OpenVSwitch, etc.

Este lenguaje es southbound lo cual implica que sirve para dar instrucciones a los nodos de red de manera complementaria o suplementaria al protocolo OpenFlow, entre los beneficios encontrados por el Consorcio de Lenguaje P4 con respecto a los procesadores de paquetes actuales están: la flexibilidad, la asignación y gestión de recursos, expresividad, ingeniería de software, desacople de la evolución del hardware y software, la biblioteca de componentes y la depuración.

Este lenguaje de programación nos permite definir el procesamiento de los paquetes es decir abrir la programación de la red, dándonos la capacidad de hacerla funcionar sin basarse en protocolos pre establecidos, esta flexibilidad es de suma importancia para la investigación y en el mercado ya hay nodos que soportan este lenguaje, como el Pica8, esta es una de las más relevantes características a considerar al momento de implementar la red.

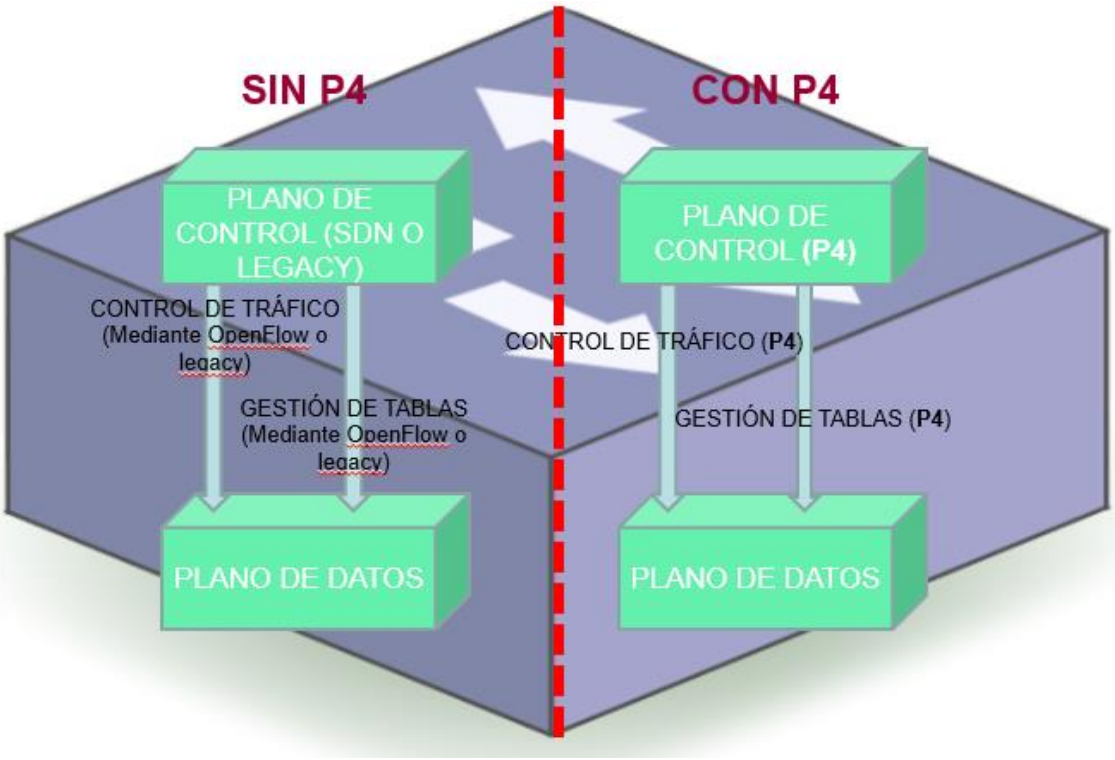


Figura 3.2 Diferencia básica entre un nodo que usa P4 vs uno que no lo usa. Fuente: Elaboración propia basada en el documento P416 Language Specification versión 1.0.0

3.3 TOPOLOGÍA DE LA PROPUESTA DE LABORATORIO SDN

La FIEE-UNMSM requiere para la formación integral de sus estudiantes, una red de laboratorio SDN, que permita que sus estudiantes y profesores comprendan la tecnología SDN y realicen innovación generando las aplicaciones de la red. Este laboratorio impulsará la investigación y con ello fomentará el intercambio de experiencias y conocimientos con otros centros de investigación nacionales e internacionales. Siendo una de las premisas en el diseño de este laboratorio el libre acceso a todos los módulos/softwares y las actualizaciones de sus nodos: Open Source. Es oportuno indicar que los principales fabricantes de equipos SDN, que son los que ofrecen al mercado equipos basados en software propietarios, son los impulsores de la comunidad de desarrolladores de la tecnología SDN en Open Source o software libre, como es el caso de la ONF.

Además es importante que en el contexto actual de Emergencia Nacional por el virus Covid-19 se entregue una propuesta con capacidad de conexión y administración remota, de modo que tanto el personal docente como los estudiantes puedan realizar / supervisar configuraciones en la red de laboratorio a distancia.

La topología propuesta se basa en dos (02) switches SDN de core/distribución, tres (03) switches SDN de conexión, un (01) controlador SDN, un (01) servidor de mediciones/base de datos donde se almacenará los datos obtenidos por el controlador SDN desde los switches, un (01) switch de gestión de la red, una (01) PC para realizar la gestión del laboratorio SDN, un (01) firewall para conectarse a este laboratorio desde Internet.

En la figura 3.3.1 se ilustra con detalle la topología del laboratorio SDN propuesta para su posterior implementación en la Facultad de Ingeniería Electrónica y Eléctrica de la Universidad Nacional Mayor de San Marcos; cumpliendo de esta manera el segundo objetivo de la tesis.

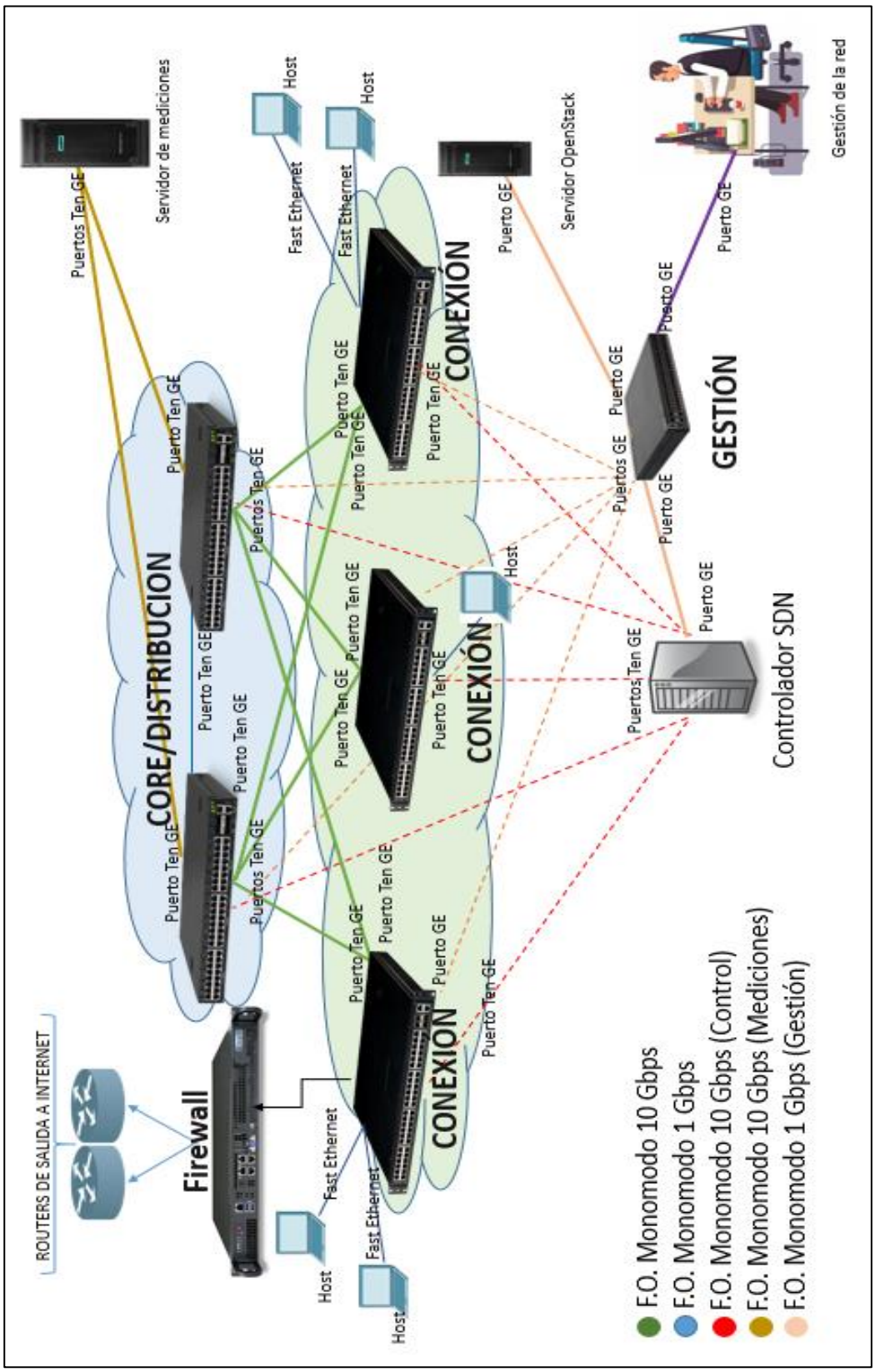


Figura 3.3.1 Diagrama topológico de la Propuesta de Laboratorio SDN (Control Plane + User Plane + Management Plane Local).

En la siguiente figura se muestra la propuesta de conexión VPN SSTL/TLS con lo cual se cumple el objetivo 4 de la tesis.

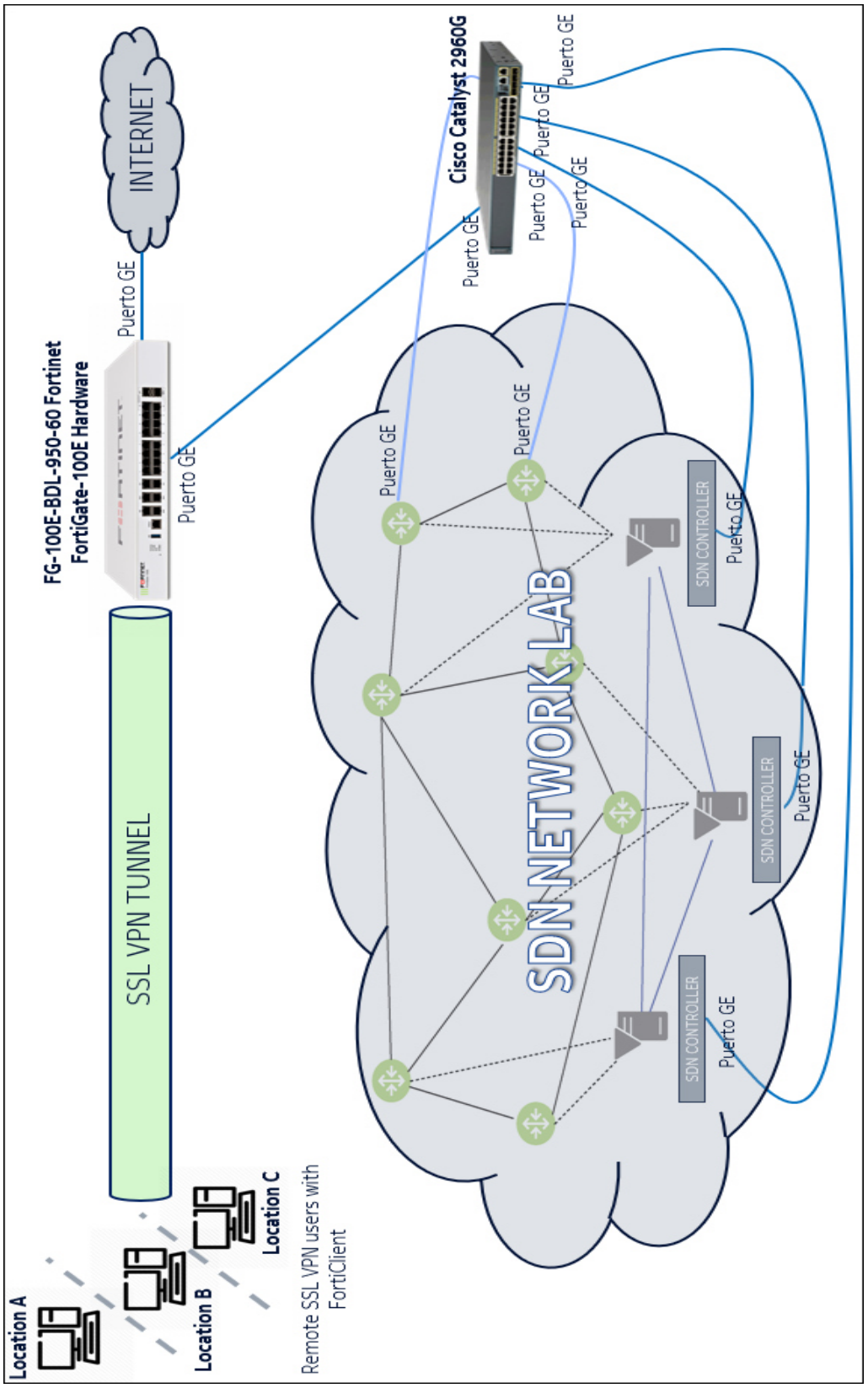


Figura 3.3.2 Diagrama topológico de la Propuesta de Laboratorio SDN (Management Plane Remoto).

3.3.1 TRANSCEPTORES OPTICOS

Son interfaces ópticas/eléctricas las cuales sirven para poder convertir los datos llevados a través de medios ópticos (fibra óptica) a señales eléctricas, las especificaciones de los estándares de los transceptores ópticos/eléctricos son definidos por la Storage Networking Industry Association (Asociación de la industria de redes de almacenamiento), la cual es una organización sin fines de lucro [80]. Las especificaciones del transceptor QSFP+ 40G se encuentran en el repositorio FTP con enlace: ftp://ftp.seagate.com/sff/SFF-8436.PDF.

3.3.2 SWITCH SDN DE CORE/DISTRIBUCIÓN

Dos (02) Switches OpenFlow que constituyen el nivel de Core/Distribución, estos equipos se conectarán entre ellos y también a los del nivel inferior (Conexión); a través de estos fluirá cualquier tráfico de la red de laboratorio.

Cada Switch debe tener 48 puertos 10 GE SFP +, 1 puerto GE de gestión y 6 puertos 40G QSFP+, con entrada de alimentación para 110v – 220 VAC. Debe tener capacidad dual stack (protocolo IPv4/IPv6). El objetivo de proponer esta cantidad de puertos es que esta red de laboratorio sea escalable, es decir, al cumplir estos switches la funcionalidad de Core/Distribución, necesitan una cantidad considerable de puertos de modo que al crecer la red a nivel de conexión, dichos nodos de conexión tengan puertos disponibles en el nivel de distribución a los cuales conectarse, de modo que en un futuro cercano a la implementación se pueda migrar todo el tráfico de la red de la Facultad de Ingeniería Electrónica y Eléctrica de la UNMSM a una SDN.

Deben soportar los protocolos OpenFlow v1.3 o superiores, NETCONF, BGP, PCEP, SNMP.

Deben contar con una garantía de al menos 5 años, redundancia en fuente de poder y ventilador. Deben poseer funcionalidad de capas 2 y 3 del modelo OSI.

3.3.3 SWITCH SDN DE CONEXIÓN

Tres (03) Switches OpenFlow que constituyen el nivel de conexión, estos equipos son los que se conectarán directamente a los equipos de usuario. Llevarán la información desde los hosts hacia el nivel de core y también de regreso.

Cada Switch debe tener 24 puertos GE, 4x10 Ten GE SFP+, 1 puerto GE de gestión, con entrada de alimentación para 110v – 220 VAC. Debe tener capacidad dual stack (protocolo IPv4/IPv6).

Deben soportar los protocolos OpenFlow v1.3 o superiores, NETCONF, BGP, PCEP, SNMP.

Deben contar con una garantía de al menos 5 años, redundancia en fuente de poder y ventilador. Debe poseer funcionalidad de capas 2 y 3 del modelo OSI.

3.3.4 CONTROLADOR SDN

El controlador debe soportar el OpenDayLight y FloodLight como mínimo, alojado en un servidor lógico con redundancia física; es decir, dos (02) servidores físicos, cada uno con una capacidad de 10 TB HDD, procesador Core I9 de 5 GHz 9na generación, arquitectura de 64 bits, memoria RAM 256 GB DDR4, controlador dual de red integrado: Dual Gigabit Ethernet, entradas HDMI y USB 3.1, tarjeta de red con dos interfaces 10 G SFP+ con soporte DPDK.

3.3.5 SERVIDOR DE MEDICIONES/BASE DE DATOS

Un (01) servidor de mediciones, donde se almacenará el tráfico de datos de la red y servirá para sacar indicadores de desempeño de la red y del comportamiento del tráfico, así como de la interacción del controlador con esta.

Este servidor debe tener una capacidad de 1 TB SSD, procesador Core I7 de 3.7 GHz, arquitectura de 64 bits, memoria RAM 256 GB DDR4, controlador dual de red integrado: Dual Gigabit Ethernet, entradas HDMI y USB 3.1, tarjeta de red con dos interfaces 10 G SFP+ con soporte DPDK.

3.3.6 SWITCH DE GESTIÓN DE LA RED

Un (01) switch convencional (legacy) con 24 puertos GE, 2 puertos Ten GE, puerto de consola, con entrada de alimentación para 110v – 220 VAC. Redundancia a nivel de tarjeta controladora, fuente de poder y ventiladores. Este Switch servirá para conectarse a la red de manera convencional y gestionarla conectándose directamente al controlador, así como al servidor OpenStack, de modo que se pueda modificar la red en todo aspecto sin involucrar físicamente la gestión.

Deben contar con una garantía de al menos 5 años, redundancia en fuente de poder y ventilador. Debe poseer funcionalidad de capas 2 y 3 del modelo OSI en modalidad dual stack (IPv4/IPv6).

Debe contar con LACP, VTP, auto negociación de puertos (half dúplex, full dúplex).

3.3.7 PC PARA GESTIÓN

Una (01) PC con capacidad mínima de 2 TB HDD, procesador Core I7 de 3.7 GHz, arquitectura de 64 bits, memoria RAM 64 GB DDR4, puerto GE, entradas HDMI y USB 3.1, de preferencia doble sistema operativo nativamente (Windows 10 y Ubuntu/CentOS). Esta PC servirá para que el administrador de la red se conecte físicamente tanto al controlador como al servidor OpenStack y pueda gestionar la red.

3.3.8 FIREWALL

Un (01) Firewall que servirá de filtro entre el flujo de datos de la red de laboratorio y la red externa, este equipo debe contar con capacidad mínima de 1 TB HDD, procesador Intel Xeon de 8 núcleos, dual gigabit ethernet, memoria RAM 8 GB DDR4, tarjeta de red con 2 interfaces 10 G SFP+ y una interfaz Dual Gigabit Ethernet, Intelligent Platform Management Interface (IPMI), que permita una conexión simultánea de más de 100 usuarios simultáneos y una velocidad de transferencia de datos mínima de 100 Mbps, una opción adecuada sería el **FG-100E-BDL-950-60** de manera que en las PCs remotas se configure el Forticlient para que accedan a la red de laboratorio tal cual estuvieran desde la intranet, esta configuración se comenta en el **Anexo "D**".

3.3.9 SERVIDOR OPEN STACK

Un (01) servidor para poder cargar estadísticas y configuraciones en la nube, debe contar con capacidad mínima de 12 TB HDD, Core I7 de 3.7 GHz, arquitectura de 64 bits, memoria RAM 256 GB DDR4, tarjeta de red con 2 interfaces 10 G SFP+ y una interfaz Dual Gigabit Ethernet, Intelligent Platform Management Interface (IPMI).

En la Tabla 3.1 se muestra un resumen de las características técnicas de los equipos de red de laboratorio SDN propuesto para FIEE de la UNMSM.

Cantidad	Equipo	Características
2	SWITCH SDN DE CORE/DISTRIBUCIÓN	48 puertos 10 GE SFP +, 1 puerto GE de gestión y 6 puertos 40G QSFP+, con entrada de alimentación para 110v – 220 VAC. Debe tener capacidad dual stack (protocolo IPv4/IPv6). Soporte de protocolos OpenFlow v1.3, NETCONF, BGP, PCEP, SNMP.
		Deben contar con una garantía de al menos 5 años, redundancia en fuente de poder y ventilador. Debe poseer funcionalidad de capas 2 y 3 del modelo OSI.
3	SWITCH SDN DE CONEXIÓN	24 GE, 4x10 Ten GE SFP+, 1 puerto GE de gestión, con entrada de alimentación para 110v – 220 VAC. Debe tener capacidad dual stack (protocolo IPv4/IPv6). Soporte de protocolos OpenFlow v1.3, NETCONF, BGP, PCEP, SNMP.
		Deben contar con una garantía de al menos 5 años, redundancia en fuente de poder y ventilador. Debe poseer funcionalidad de capas 2 y 3 del modelo OSI.
2	CONTROLADOR SDN	Capacidad de 10 TB HDD, procesador Core I9 de 5 GHz 9na generación, arquitectura de 64 bits, memoria RAM 256 GB DDR4, controlador dual de red integrado: Dual Gigabit Ethernet, entradas HDMI y USB 3.1, tarjeta de red con dos interfaces 10 G SFP+ con soporte DPDK.
1	SERVIDOR DE MEDICIONES/BASE DE DATOS	capacidad de 1 TB SSD, procesador Core I7 de 3.7 GHz, arquitectura de 64 bits, memoria RAM 256 GB DDR4, controlador dual de red integrado: Dual Gigabit Ethernet, entradas HDMI y USB 3.1, tarjeta de red con dos interfaces 10 G SFP+ con soporte DPDK.
1	SWITCH DE GESTIÓN DE LA RED	24 puertos GE, 2 puertos Ten GE, puerto de consola, con entrada de alimentación para 110v – 220 VAC. Redundancia a nivel de tarjeta controladora, fuente de poder y ventiladores.
		Deben contar con una garantía de al menos 5 años, redundancia en fuente de poder y ventilador. Debe poseer funcionalidad de capas 2 y 3 del modelo OSI en modalidad dual stack (IPv4/IPv6). Protocolos LACP, VTP, auto negociación de puertos (half dúplex, full dúplex).
1	PC PARA GESTIÓN	2 TB HDD, procesador Core I7 de 3.7 GHz, arquitectura de 64 bits, memoria RAM 64 GB DDR4, puerto GE, entradas HDMI y USB 3.1, de preferencia doble sistema operativo nativamente (Windows 10 y Ubuntu/CentOS)
1	FIREWALL	1 TB HDD, procesador Intel Xeon de 8 nucleos, dual gigabit ethernet, memoria RAM 8 GB DDR4, tarjeta de red con 2 interfaces 10 G SFP+ y una interfaz Dual Gigabit Ethernet, Intelligent Platform Management Interface (IPMI).
1	SERVIDOR OPEN STACK	12 TB HDD, Core I7 de 3.7 GHz, arquitectura de 64 bits, memoria RAM 256 GB DDR4, tarjeta de red con 2 interfaces 10 G SFP+ y una interfaz Dual Gigabit Ethernet, Intelligent Platform Management Interface (IPMI).

Tabla 3.1 Resumen de Equipos de Red de Laboratorio SDN.

3.4 ASPECTOS ECONÓMICOS DE LA PROPUESTA DE LABORATORIO SDN

En la Tabla 3.2 se muestra un resumen de los costos de los equipos de red de laboratorio SDN propuesto para FIEE de la UNMSM.

Cantidad	Equipo	Características	Costo x unidad	Costo Total
2	SWITCH SDN DE CORE/DISTRIBUCIÓN	48 puertos 10 GE SFP +, 1 puerto GE de gestión y 6 puertos 40G QSFP+, con entrada de alimentación para 110v – 220 VAC. Debe tener capacidad dual stack (protocolo IPv4/IPv6). Soporte de protocolos OpenFlow v1.3, NETCONF, BGP, PCEP, SNMP. Deben contar con una garantía de al menos 5 años, redundancia en fuente de poder y ventilador. Debe poseer funcionalidad de capas 2 y 3 del modelo OSI.	S/ 30,000	S/ 60,000
2	LICENCIA DE SWITCH SDN CORE	Licencia de software para Switch SDN Core/Distribución	S/ 10,000	S/ 20,000
3	SWITCH SDN DE CONEXIÓN	24 GE, 4x10 Ten GE SFP+, 1 puerto GE de gestión, con entrada de alimentación para 110v – 220 VAC. Debe tener capacidad dual stack (protocolo IPv4/IPv6). Soporte de protocolos OpenFlow v1.3, NETCONF, BGP, PCEP, SNMP. Deben contar con una garantía de al menos 5 años, redundancia en fuente de poder y ventilador. Debe poseer funcionalidad de capas 2 y 3 del modelo OSI.	S/ 6,000	S/ 18,000
3	LICENCIA DE SWITCH SDN CONEXIÓN	Licencia de software para Switch SDN Conexión	S/ 4,000	S/ 12,000
2	CONTROLADOR SDN	Capacidad de 10 TB HDD, procesador Core I9 de 5 GHz 9na generación, arquitectura de 64 bits, memoria RAM 256 GB DDR4, controlador dual de red integrado: Dual Gigabit Ethernet, entradas HDMI y USB 3.1, tarjeta de red con dos interfaces 10 G SFP+ con soporte DPDK.	S/ 60,000	S/ 120,000
1	SERVIDOR DE MEDICIONES/BASE DE DATOS	capacidad de 1 TB SSD, procesador Core I7 de 3.7 GHz, arquitectura de 64 bits, memoria RAM 256 GB DDR4, controlador dual de red integrado: Dual Gigabit Ethernet, entradas HDMI y USB 3.1, tarjeta de red con dos interfaces 10 G SFP+ con soporte DPDK.	S/ 45,000	S/ 45,000
1	SWITCH DE GESTIÓN DE LA RED	24 puertos GE, 2 puertos Ten GE, puerto de consola, con entrada de alimentación para 110v – 220 VAC. Redundancia a nivel de tarjeta controladora, fuente de poder y ventiladores. Deben contar con una garantía de al menos 5 años, redundancia en fuente de poder y ventilador. Debe poseer funcionalidad de capas 2 y 3 del modelo OSI en modalidad dual stack (IPv4/IPv6). Protocolos LACP, VTP, auto negociación de puertos (half dúplex, full dúplex).	S/ 2,500	S/ 2,500
1	PC PARA GESTIÓN	2 TB HDD, procesador Core I7 de 3.7 GHz, 9na generación, arquitectura de 64 bits, memoria RAM 64 GB DDR4, puerto GE, entradas HDMI y USB 3.1, de preferencia doble sistema operativo nativamente (Windows 10 y Ubuntu/CentOS)	S/ 8,000	S/ 8,000
1	FIREWALL	1 TB HDD, procesador Intel Xeon de 8 nucleos, dual gigabit ethernet, memoria RAM 8 GB DDR4, tarjeta de red con 2 interfaces 10 G SFP+ y una interfaz Dual Gigabit Ethernet, Intelligent Platform Management Interface (IPMI).	S/ 10,000	S/ 10,000
2	SERVIDOR OPEN STACK	12 TB HDD, Core I7 de 3.7 GHz, arquitectura de 64 bits, memoria RAM 256 GB DDR4, tarjeta de red con 2 interfaces 10 G SFP+ y una interfaz Dual Gigabit Ethernet, Intelligent Platform Management Interface (IPMI).	S/ 45,000	S/ 90,000
1	SERVICIOS DE INSTALACIÓN Y CABLES	Servicios de instalación, configuración y costo de cables.	S/ 40,000	S/ 40,000
TOTAL	S/ 425,500			

Tabla 3.2 Costos de Equipos de Red de Laboratorio SDN.

3.5 ALTERNATIVAS PROPIETARIAS DE UN LABORATORIO SDN

Cisco: Cisco ofrece una solución SDN a través de sus switches Nexus, mediante el plugin OpenFlow Agent, también a través de sus ASR9000 y los switches Catalyst, usando el Cisco Open SDN Controller 1.2 basado en OpenDayLight, dependiendo de las características de la red a implementar y junto con ellos van determinadas licencias como las del controlador, en el cual se otorga una licencia base que permite la conexión hasta de 3 controladores en forma de cluster, y para el control de nodos se pueden comprar licencias de 50, 100, 250, 500 o 1000 nodos las cuales pueden tener vigencias de 1, 3 y 5 años.

Juniper: Juniper ofrece una solución SDN con su controlador NorthStar Controller, el cual soporta ISIS-TE, BGP-LS, NETCONF, REST CONF, PCEP, Path Control Element, entre otros, tiene capacidad de desfragmentación de la red con ingeniería de tráfico y optimización global de LSP's (Label Switch Paths),

En cuanto a sus switches los que soportan OpenFlow son los de la serie QFX5100 y EX4600, se les debe instalar un paquete de software Openflow. Al igual que en Cisco se deben comprar licencias dependiendo de la cantidad de nodos que se desee gestionar.

Huawei: Huawei ofrece un controlador basado en ONOS, llamado Huawei Agile Controller el cual permite control de QoS, uso de analítica de big data para descubrir potenciales ataques y un Guest Manager similar a un portal cautivo.

También ofrece el Service Orquestation Manager el cual tiene la capacidad de virtualizar dispositivos de seguridad gestionando Access Control Lists (ACL's) y Access Control Lists basadas en usuarios (UCL's).

Dentro de estas tres soluciones privadas tenemos funcionalidades interesantes en cuanto a seguridad y gestión, sin embargo, el acceso al código es limitado y cada funcionalidad tiene un costo asociado, así como el acceso a la gestión de un número mayor de nodos, todo esto hace que estas soluciones no sean las más óptimas para un entorno de investigación y formación de futuros especialistas de telecomunicaciones, el cual es nuestro caso de estudio, pero sí son óptimas para una red en producción.

CAPÍTULO IV

SIMULACIÓN DEL LABORATORIO SDN

4.1 INTRODUCCIÓN

Habiendo seleccionado software Open Source como base del controlador SDN y existiendo herramientas de libre acceso como Mininet y Wireshark, se va a simular y analizar el escenario propuesto para el laboratorio SDN de la Facultad de Ingeniería Electrónica y Eléctrica de la UNMSM haciendo uso de ambas herramientas. Mininet es un emulador de redes definidas por software que permite interconectar controladores SDN, conmutadores OpenFlow y usuarios finales. Una de las ventajas del uso de Mininet es de estar implementado en código abierto y ofrece una fácil capacidad de implementar diversas topologías de redes definidas por software, soportando un conjunto de controladores como OpenDayLight, Floodlight y Ryu. Mininet tiene una herramienta denominado Miniedit que permite crear topologías mediante una interfaz gráfica en lugar de utilizar líneas de comandos (CLI). Mininet también da la opción de crear topologías utilizando scripts en Python. Otra facilidad de Mininet es de ofrecer un escenario para la verificación a los desarrolladores de aplicación para la gestión de SDN.

4.2 SIMULACIÓN DE LA TOPOLOGÍA PROPUESTA DE LABORATORIO SDN

4.2.1 Recursos informáticos utilizados

La simulación del escenario del laboratorio propuesto se ha realizado en una computadora personal que tiene 12 GB de RAM, procesador Intel Core I5-4210 de 2.4GHz, Windows 8.1 de 64 bits y una memoria libre de 500 GB.

En la tabla 4.1 se resumen estas características.

Características	Valores del computador físico	Valores de la Máquina Virtual que contiene Mininet	Valores de la Máquina Virtual que contiene Miniedit
CPU	Intel Core I5-4210 de 2.4GHz	1 Core	1 Core
Memoria RAM	12 GB	2 GB	2 GB
Memoria en Disco Duro disponible	500 GB	40 GB	40GB
Sistema Operativo	Windows 8.1 – 64 Bits	Ubuntu 14.04 LTS – 32 Bits	Ubuntu 14.04 LTS – 32 Bits

Tabla 4.1 Características de la PC donde se simula la red propuesta

4.2.2 Simulación de la topología propuesta de laboratorio SDN

Considerando la topología propuesta en el capítulo anterior, figura 3.3, tenemos los hosts “h1” y “h2”, conectados mediante OpenVSwitches para nivel Core, Distribución y Conexión como el caso anterior y siendo controlados por el controlador “c0” el cual es un controlador OpenDayLight con dirección IP: 192.168.1.41 con el puerto OpenFlow 6633 y protocolo OpenFlow 1.3. Si bien el software Mininet ofrece topologías predefinidas, esta fue una topología personalizada. A continuación, en la Figura 4.1 se muestra la topología simulada.

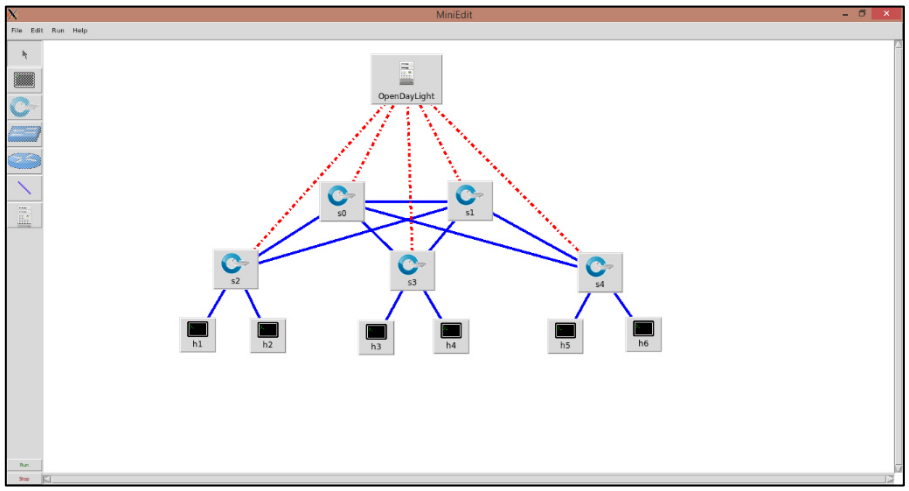


Figura 4.1 Imagen de la red de laboratorio propuesta para la FIEE-UNMSM simulada bajo la arquitectura SDN en el software Mininet

Para poder acceder al entorno gráfico del Mininet, se activó el editor gráfico Miniedit usando el comando "`sudo ~/mininet/examples/miniedit.py`" en la máquina virtual que contiene el software Mininet.

A continuación, en la Figura 4.2 se muestra el terminal de la máquina virtual, la cual contiene al software Mininet.

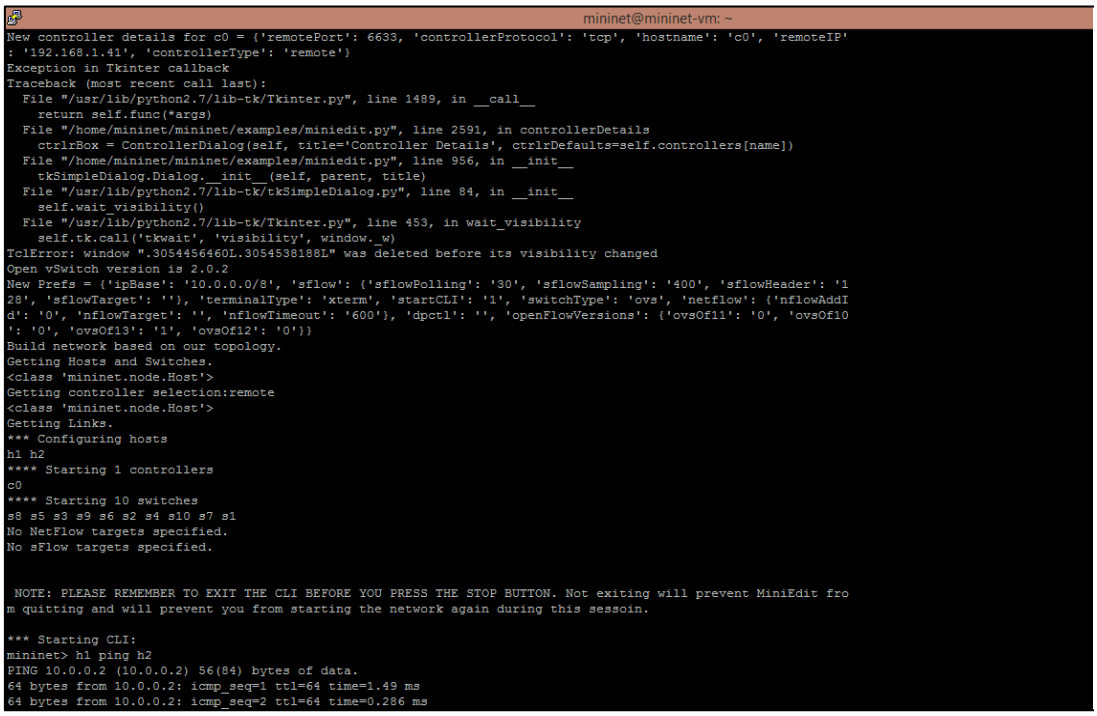


Figura 4.2. Command line interface de la máquina virtual que contiene al software Mininet.

Se tiene el siguiente script en Python con el cual se puede replicar la red simulada, cargándolo en el editor gráfico o directamente en el Shell de la máquina virtual, teniendo la IP de controlador: 192.168.1.41 y puerto 6633. Además, IP's de los hosts virtuales pertenecientes al segmento de red privado 10.0.0.0/8.

```
#!/usr/bin/python

from mininet.net import Mininet
from mininet.node import Controller, RemoteController, OVSController
from mininet.node  import CPULimitedHost, Host, Node
from mininet.node import OVSKernelSwitch, UserSwitch
from mininet.node import IVSSwitch
from mininet.cli import CLI
from mininet.log import setLogLevel, info
from mininet.link import TCLink, Intf
from subprocess import call

def myNetwork():

   net = Mininet( topo=None,
              build=False,
              ipBase='10.0.0.0/8')

   info( '*** Adding controller\n' )
   OpenDayLight=net.addController(name='OpenDayLight',
                 controller=RemoteController,
                 ip='192.168.1.41',
                 protocol='tcp',
                 port=6633)

   info( '*** Add switches\n')
   s4 = net.addSwitch('s4', cls=OVSKernelSwitch)
   s1 = net.addSwitch('s1', cls=OVSKernelSwitch)
   s0 = net.addSwitch('s0', cls=OVSKernelSwitch)
   s2 = net.addSwitch('s2', cls=OVSKernelSwitch)
   s3 = net.addSwitch('s3', cls=OVSKernelSwitch)

   info( '*** Add hosts\n')
   h3 = net.addHost('h3', cls=Host, ip='10.0.0.3', defaultRoute=None)
   h4 = net.addHost('h4', cls=Host, ip='10.0.0.4', defaultRoute=None)
   h5 = net.addHost('h5', cls=Host, ip='10.0.0.5', defaultRoute=None)
   h6 = net.addHost('h6', cls=Host, ip='10.0.0.6', defaultRoute=None)
   h1 = net.addHost('h1', cls=Host, ip='10.0.0.1', defaultRoute=None)
   h2 = net.addHost('h2', cls=Host, ip='10.0.0.2', defaultRoute=None)
   info( '*** Add links\n')
   net.addLink(h6, s4)
   net.addLink(s2, s0)
   net.addLink(s0, s1)
   net.addLink(s1, s4)
   net.addLink(s0, s3)
   net.addLink(s1, s3)
   net.addLink(s0, s4)
   net.addLink(s1, s2)
   net.addLink(h1, s2)
   net.addLink(h2, s2)
   net.addLink(h3, s3)
   net.addLink(h4, s3)
   net.addLink(h5, s4)

   info( '*** Starting network\n')
   net.build()
   info( '*** Starting controllers\n')
```

```
    for controller in net.controllers:
        controller.start()

    info( '*** Starting switches\n')
    net.get('s4').start([OpenDayLight])
    net.get('s1').start([OpenDayLight])
    net.get('s0').start([OpenDayLight])
    net.get('s2').start([OpenDayLight])
    net.get('s3').start([OpenDayLight])

    info( '*** Post configure switches and hosts\n')

    CLI(net)
    net.stop()

if __name__ == '__main__':
    setLogLevel( 'info' )
    myNetwork()
```

A continuación, se muestra la línea de comandos de la máquina virtual que contiene al controlador OpenDayLight, luego de iniciar el controlador, este se ilustra en la figura 4.3.

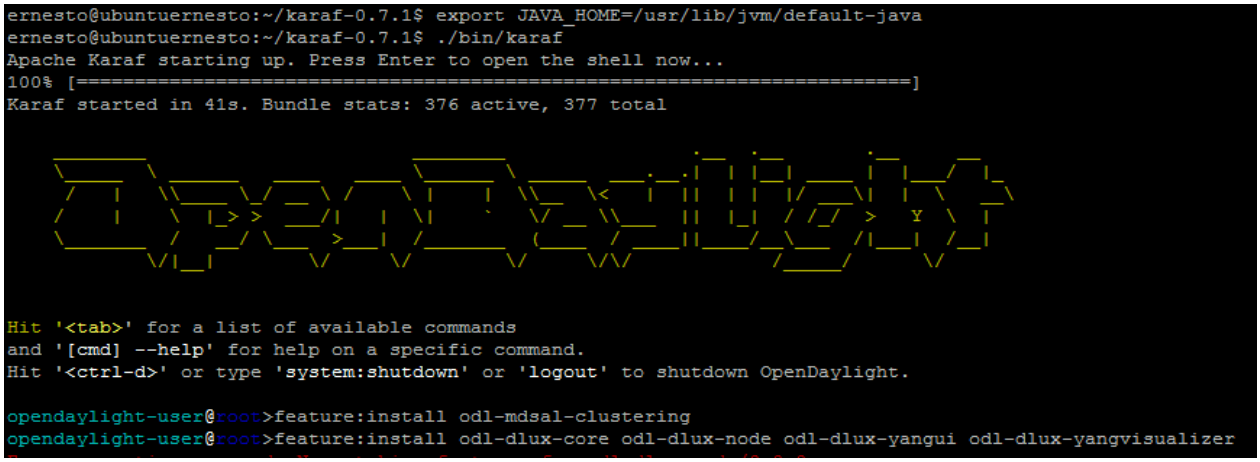


Figura 4.3. Command line interface de la máquina virtual que contiene al controlador OpenDayLight.

Se activó el *feature* que permite ver la red a través de la interfaz web ingresando el comando "feature: install odl-dlux-yangvisualizer"; entre otros necesarios para el uso. A continuación, en la Figura 4.4 se muestra la topología descubierta por el controlador OpenDayLight, accediendo a través de la interfaz web.

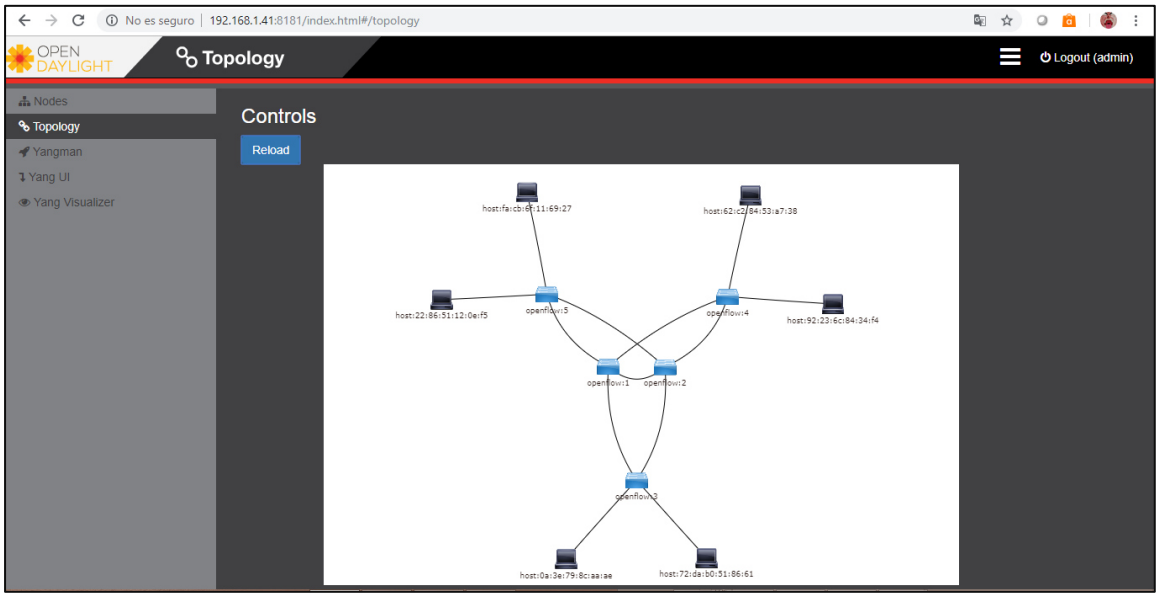


Figura 4.4 Topología vista bajo la interfaz web del controlador OpenDayLight.

A continuación, en la Figura 4.5 se muestra el terminal del host “h1”, el cual posee la IP 10.0.0.1/8, pertenece a la topología SDN dentro del Mininet y se ejecuta un ping continuo a la IP 10.0.0.6/8 la cual pertenece al host “h6”.

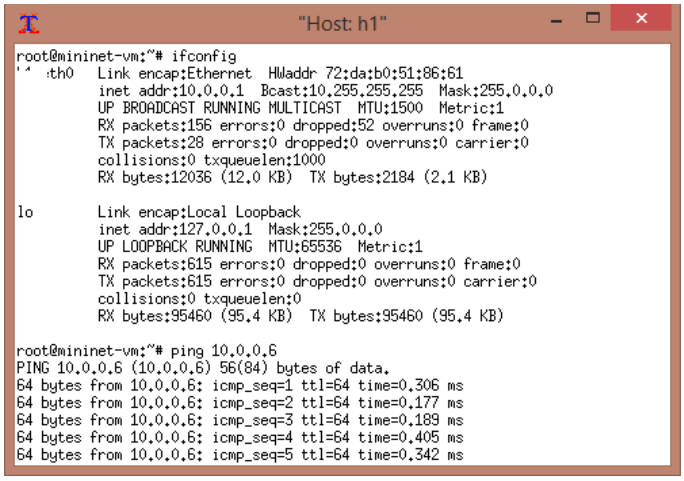


Figura 4.5 Command Line Interface del host “h1”.

Realizaremos dos pruebas esenciales para tener una referencia clara del comportamiento de la red propuesta de laboratorio bajo la arquitectura SDN, la primera la cual llamaremos “Escenario A”, la que consistirá en desactivar un enlace, con el objetivo de medir el tiempo de respuesta en capa 2, en reconvergencia y una segunda prueba la cual llamaremos “Escenario B” la cual consistirá en medir los parámetros de desempeño de esta red simulada

utilizando la herramienta IPERF y además comparar su respuesta tanto en los protocolos TCP/UDP respecto de IPv4/IPv6.

4.2.3 Escenario A propuesto: Reconvergencia

En la simulación, el camino principal es el superior (enlace entre “S1” y “S4”), por lo cual se desactiva el enlace entre el Ovswitch1 y el Ovswitch4; como se observa en la figura 4.6.

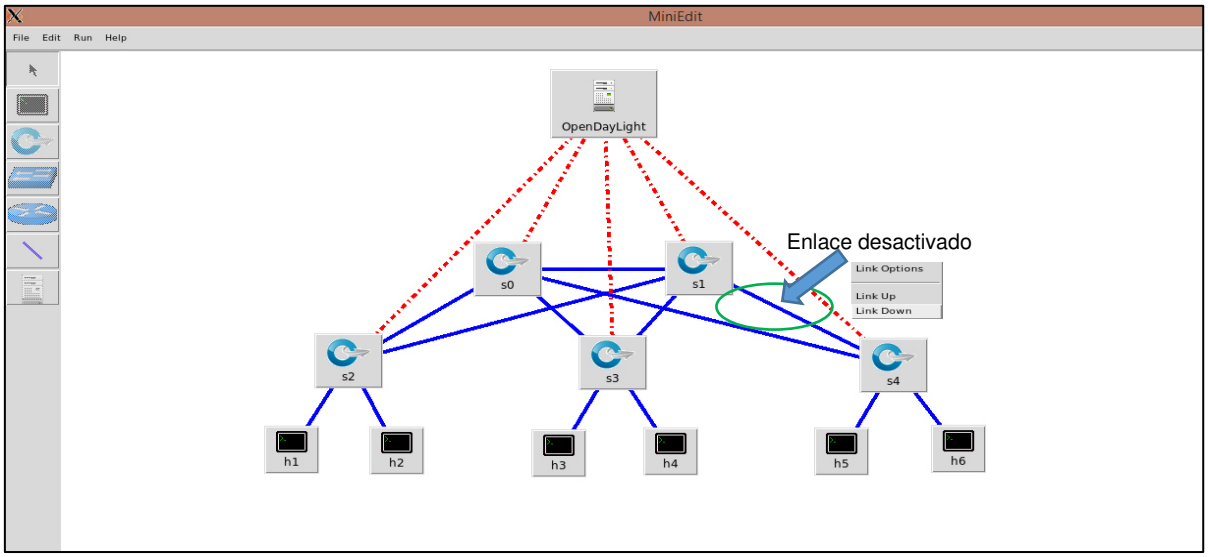


Figura 4.6 Imagen del enlace a desactivar.

A continuación, se presenta el cuadro con los resultados de las pruebas.

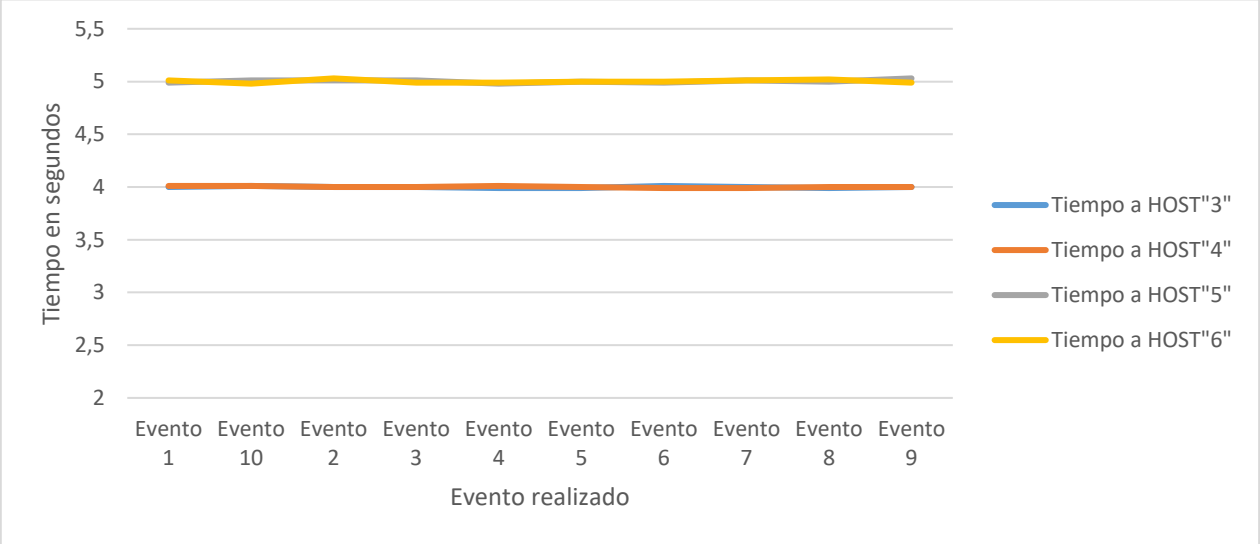


Figura 4.7 Imagen de los eventos realizados

El tiempo de reconvergencia fue aproximadamente 5 segundos, este evento fue realizado 10 veces en IPv4 y el tiempo se calculó capturando los paquetes ICMP en Wireshark, para luego comparar los tiempos del ICMP Request hasta la primera respuesta. A continuación, en la figura 4.8 se muestra la captura de los paquetes durante la caída de enlace y la reconvergencia.

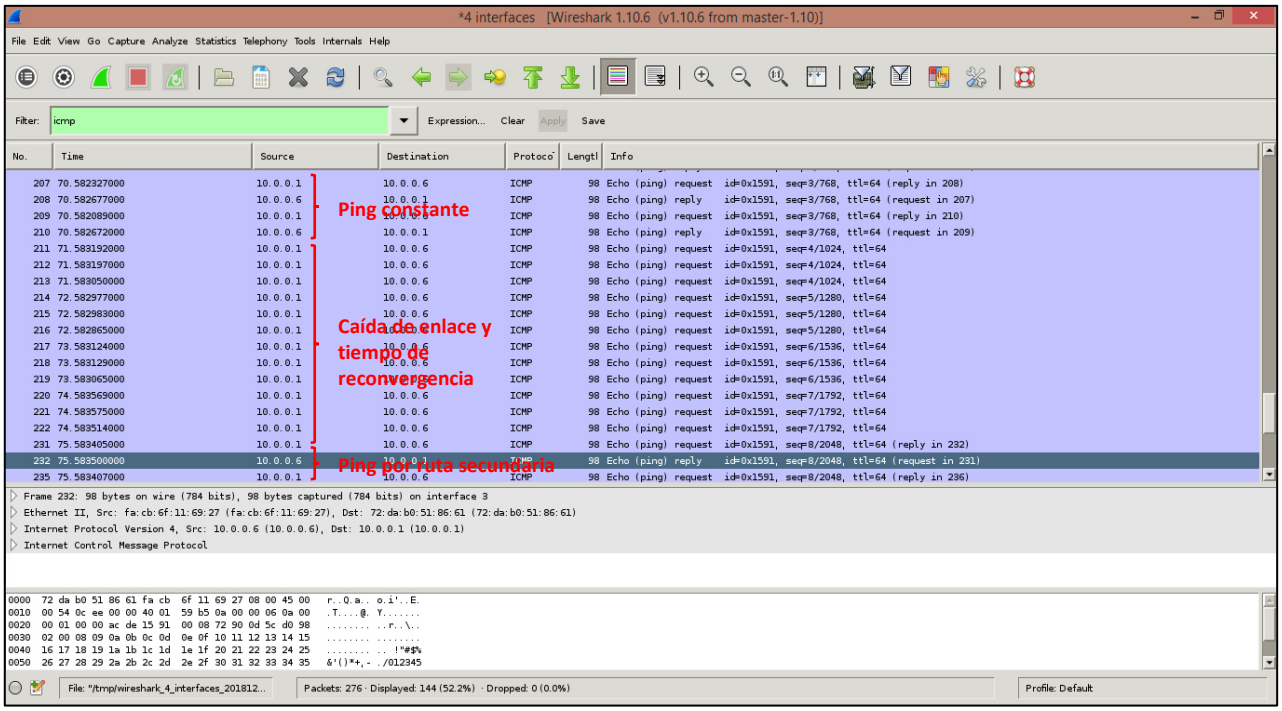


Figura 4.8 Captura en Wireshark de Paquetes ICMP de la red de laboratorio propuesta para la FIEE-UNMSM simulada en Mininet.

4.2.4 Escenario B propuesto: Throughput(Tasa de bits) y latencia

Esta prueba como se mencionó anteriormente, se realiza utilizando la herramienta IPERF desde el host 1 hacia el host 6, en diferentes casos con el objetivo de medir el desempeño de la red en el entorno de simulación, por consiguiente se analizarán 8 casos, 4 casos con IPv4 y 4 casos con IPv6, dentro del grupo de IPv4 e IPv6 se harán dos casos TCP y dos casos UDP puesto que se realizarán variaciones en los parámetros de cada protocolo, el uso de esta herramienta es muy común para medir throughput máximo alcanzado, por ejemplo al implementar una nueva tecnología, ya que se tiene que comprobar y/o averiguar los niveles que alcanza en latencia, throughput y pérdida de paquetes lo cual forma parte de su desempeño, ya sea redes móviles, de datos, de transporte, core, “end to end”, etc. A continuación, se muestran los resultados de los diferentes casos realizados:

4.2.4.1 Mediciones utilizando protocolo IPv4

•Caso 1: Prueba con configuraciones **por defecto**, es decir, uso del protocolo **TCP** y un Windows size: 86.3 KB, en la siguiente imagen (Figura 4.9) se muestra la configuración del host 1 como servidor IPERF y el puerto usado, donde el throughput máximo alcanzado es de 97.8 Mbits/sec:

* El –s implica que funcione como servidor, el –p indica el puerto y el –i es el intervalo de cada cuántos segundos se va a tomar la medición.

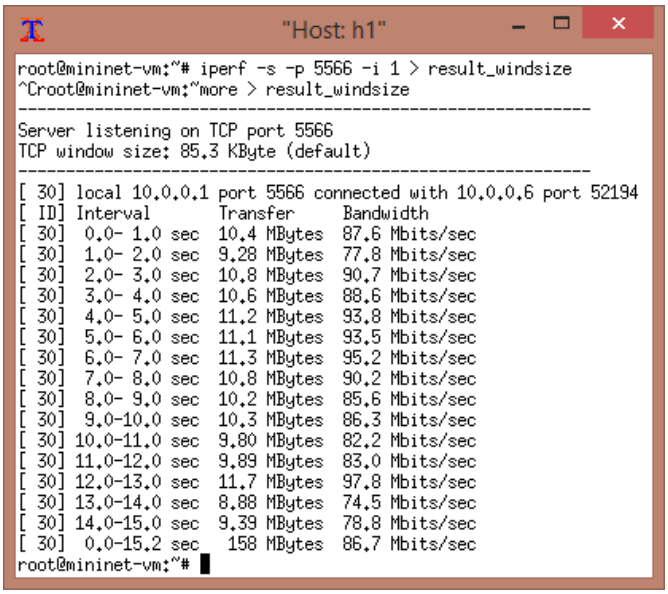


Figura 4.9 Imagen de la configuración del “Host: h1” como servidor de la herramienta IPERF y valores de medición.

En la siguiente imagen (Figura 4.10) se observa la configuración del host 6 como cliente y el ancho de banda promedio obtenido:

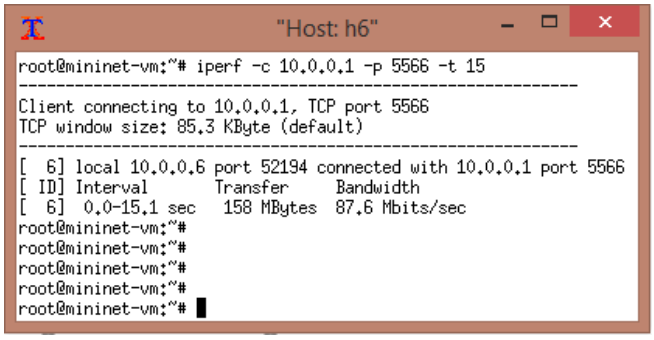


Figura 4.10 Imagen de la configuración del “Host: h6” como cliente de la herramienta IPERF.

Los resultados obtenidos se graficaron mediante el editor gráfico GNUPLOT en Linux, para lo cual previamente se acomodó los resultados en dos columnas, como se observa en la figura 4.11.

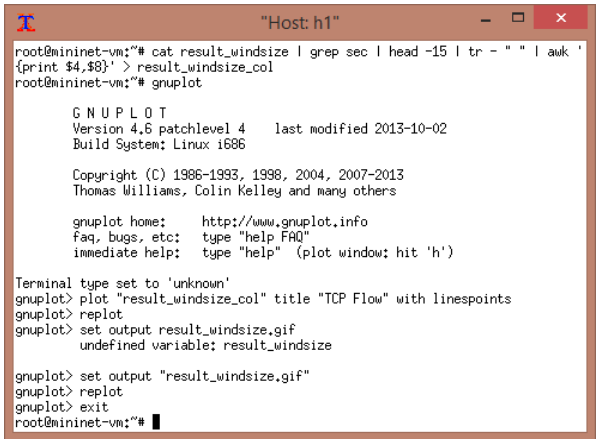


Figura 4.11 Imagen de la configuración en el "Host: h1" usando el editor gráfico GNUPLOT.

De esta simulación, se obtiene el siguiente gráfico de desempeño (Figura 4.12):

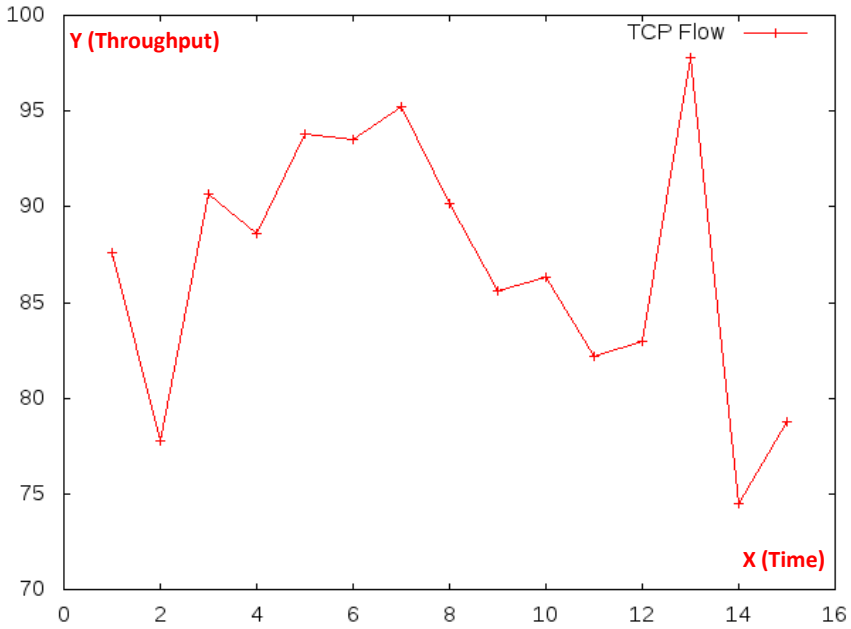


Figura 4.12 Imagen de la respuesta en throughput vs tiempo para la simulación de red de laboratorio SDN, caso 1 usando la herramienta IPERF y el editor gráfico GNUPLOT.

Además, se realizó una prueba de ping para luego ajustar el *windows size* al valor más óptimo; la simulación se hizo con enlaces de 1Gbps.

A continuación, se muestra la imagen (Figura 4.13) de la prueba realizada y el resultado obtenido:

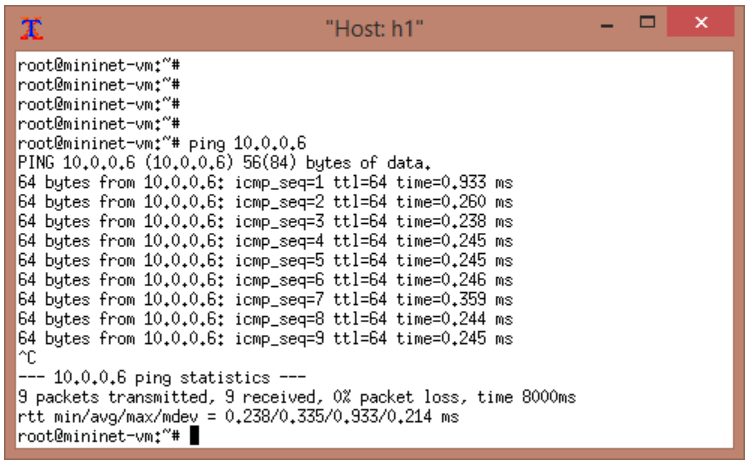


Figura 4.13. Imagen de la prueba de conectividad entre el “host 1” y el “host 6”.

Como se observa se enviaron nueve paquetes, de los cuales en el primer caso el *round trip time* es mayor puesto que aún no se tienen tablas de flujo que especifiquen cómo reenviar el paquete, luego de que el controlador actualice las tablas de flujo, los demás paquetes tienen una ruta establecida y la siguen de manera directa. Por ello se consideró los siguientes ocho paquetes para los cuales el tiempo promedio es:

$$t_{promedio} = \frac{0.260 + 0.238 + 0.245 + 0.245 + 2.46 + 0.359 + 0.244 + 0.245}{8}$$

$$= 0.26025\ ms$$

Pero, además, la ventana óptima para TCP depende del *round trip delay* y del tamaño del enlace en MB/s, con lo cual se espera una variación dependiente del cambio del *Windows size*, por lo tanto, como los enlaces son de 1Gbps (1024/8 Mbps) y el delay promedio es 0.26025 ms (0.2605 x 10^{-3} s):

$$TCP_{windows\ size\ (óptimo)} = \frac{1x1024x0.26025x10^{-3}}{8} = 33.312KB$$

•<u>Caso 2</u>: Prueba en **TCP variando** el ***windows size*** al valor de 33.312 KB tomando como valor aproximado 33.2 KB:

Con esta configuración y configurando el host “h1” como servidor, obtenemos los resultados que se muestran en el siguiente gráfico (Figura 4.14) siendo el pico máximo 97.9 Mbps:

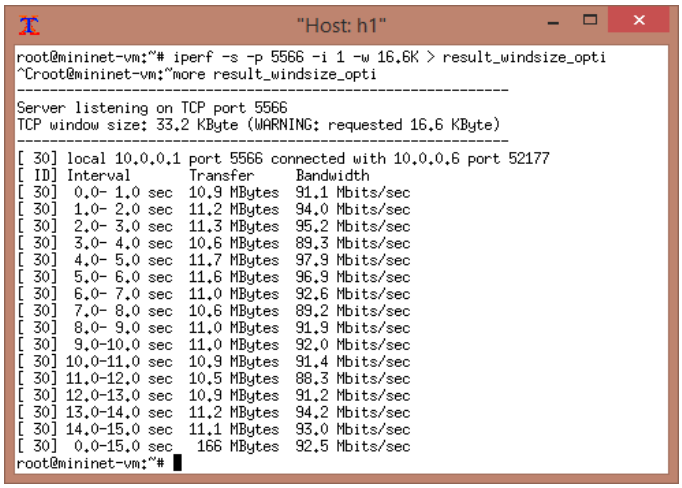


Figura 4.14 Imagen de los resultados prueba TCP con IPERF cambiando el Windows size a 33.2 KB

Como podemos observar la capacidad de througput es ligeramente superior, siendo 92.5 Mbps respecto del caso anterior que fue 86.7 Mbps. Con el gráfico 4.11 y 4.14 podemos darnos cuenta que los picos en el segundo caso (windows size adecuado) son más altos y los niveles mínimos son superiores también.

Para este caso, el desempeño se puede observar en el siguiente gráfico (Figura 4.15):

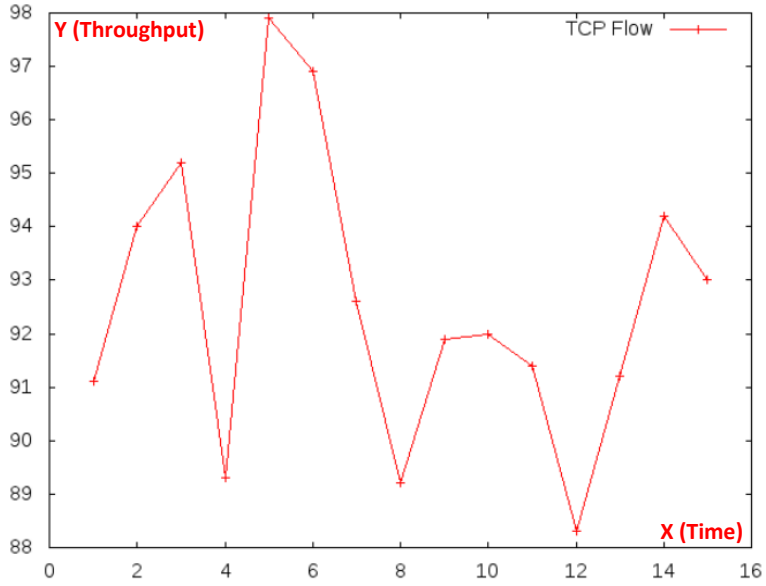


Figura 4.15 Imagen de la respuesta en throughput vs tiempo para la simulación de red de laboratorio SDN, caso 2 usando la herramienta IPERF y el editor gráfico GNUPLOT.

•Caso 3: Prueba en **UDP** con un buffer size **por defecto** y una configuración de tasa de envío de **1Gbps**, en la siguiente imagen (Figura 4.16) se observan los resultados del lado del cliente, es decir, el host “h6”:

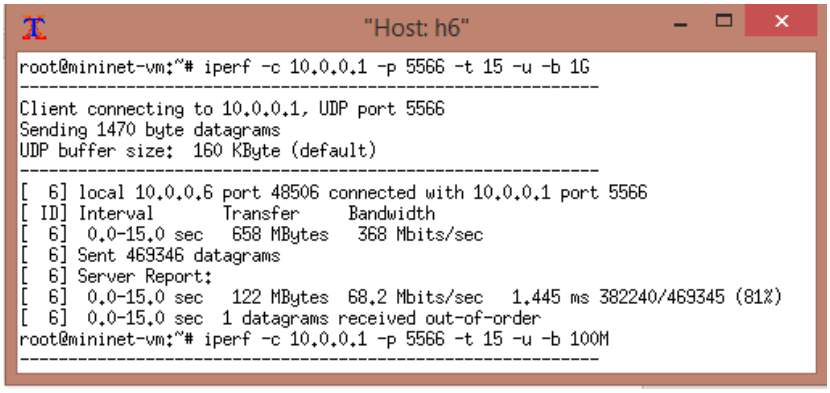


Figura 4.16. Imagen de los resultados prueba UDP con IPERF en el lado del cliente, usando una tasa de envío de 1Gbps.

En la imagen anterior se observa que el bit rate se limitó a 368 Mbit/sec y además hubo pérdida de paquetes promedio de 81%, esto debido a que la prueba se realizó con una tasa de envío superior a la soportada por la red. El hecho de enviar un bit/rate superior al soportado por la red provoca buffering y pérdida de paquetes.

En la siguiente imagen (Figura 4.17) se observan los resultados de manera detallada, del lado del servidor, es decir, host “h1”:

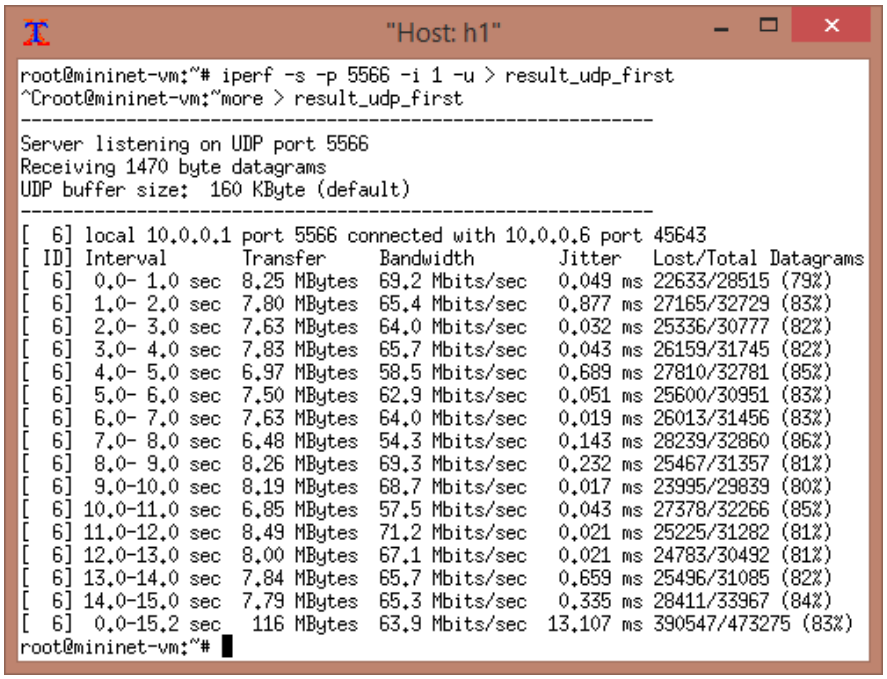


Figura 4.17. Imagen de los resultados prueba UDP con IPERF en el lado del servidor, usando una tasa de envío de 1Gbps.

De esta simulación, se obtiene el siguiente gráfico (Figura 4.18) de desempeño:

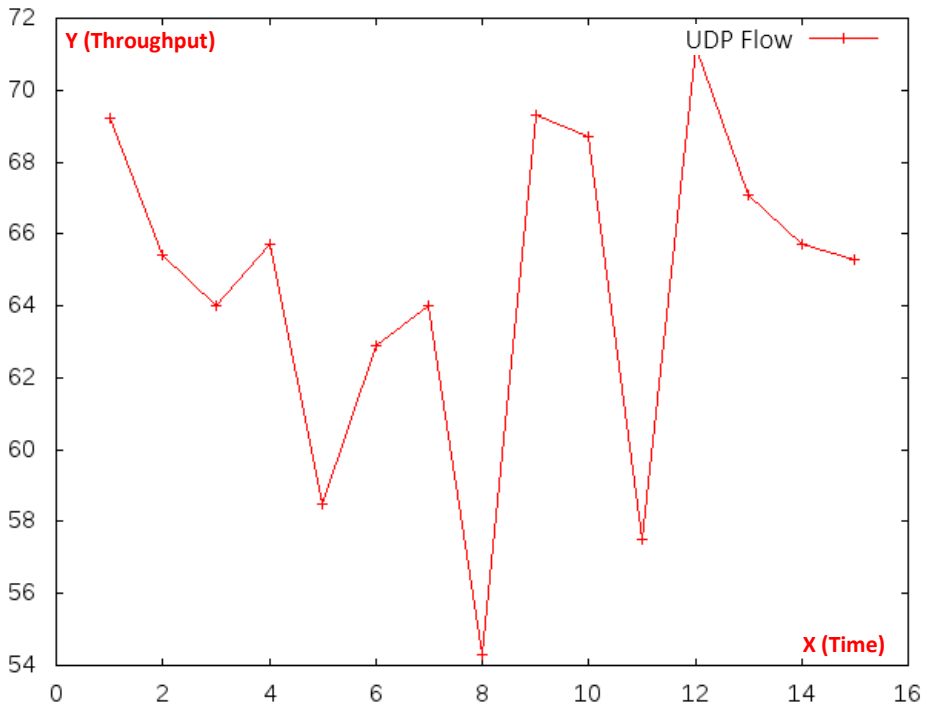


Figura 4.18. Imagen de la respuesta en throughput vs tiempo para la simulación de red de laboratorio SDN, caso 3 usando la herramienta IPERF y el editor gráfico GNUPLOT.

Sin embargo, como la herramienta se limitó a 368 Mbits/sec y se perdieron paquetes, se analizará un nuevo caso dentro del desempeño soportado por la red.

•**Caso 4**: Prueba en **UDP** con un buffer size por defecto y una configuración de tasa de envío de **100 Mbps (dentro de lo soportado)**, en la siguiente imagen (Figura 4.19) se puede observar los valores obtenidos a detalle:

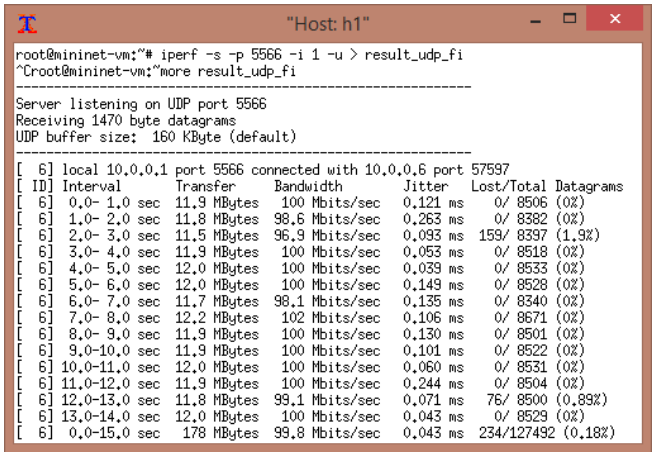


Figura 4.19. Imagen de los resultados prueba UDP con IPERF en el lado del servidor, usando una tasa de envío de 100 Mbps.

De esta prueba observamos que tenemos una pérdida de paquetes baja (0.18%) y el gráfico de desempeño se observa en la siguiente imagen (Figura 4.20):

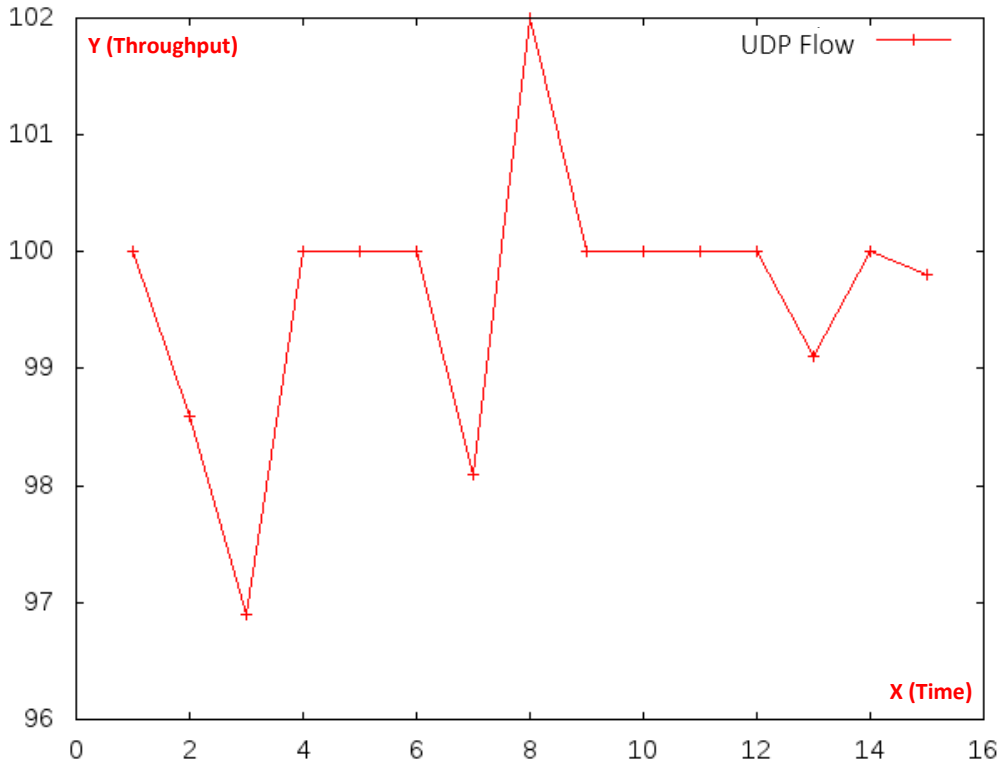


Figura 4.20. Imagen de la respuesta en throughput vs tiempo para la simulación de red de laboratorio SDN, caso 4 usando la herramienta IPERF y el editor gráfico GNUPLOT.

Con esta imagen se observa que los picos mínimos y máximos son muy superiores cuando se toma un bit/rate dentro del soportado, del mismo modo ya no se pierde paquetes e incluso llega a ser constante en determinados intervalos de tiempo.

Luego pasamos a realizar las pruebas en IPv6: Cabe mencionar que por defecto viene desactivado el IPv6 en la VM que contiene a la herramienta Mininet, el procedimiento para activar el IPv6 y poder configurarlo en los hosts virtuales se explica en el **Anexo C**

4.2.4.2 Utilizando protocolo IPv6

•Caso 5: Prueba con configuraciones **TCP por defecto** e **IPv6**, es decir, un windows size de 85.3 KB, las cuales se observan en la figura siguiente (Figura 4.21) con un pico máximo de 159 Mbit/sec:

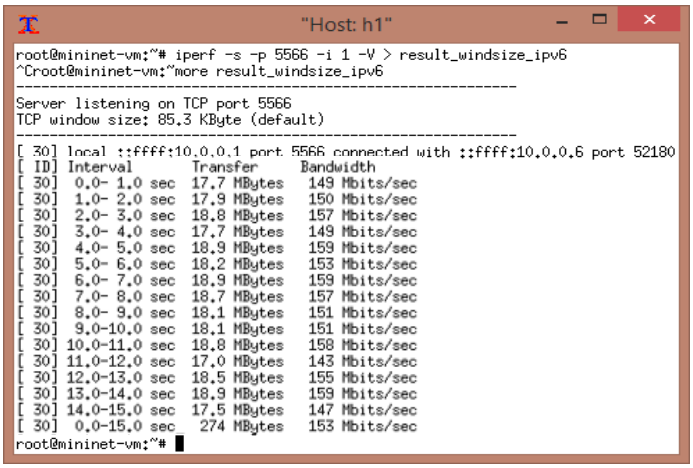


Figura 4.21. Imagen de los resultados prueba TCP con IPERF en IPv6 en el lado del servidor o host 1.

En la siguiente imagen (Figura 4.22) se observa la configuración del host 6 como cliente y el ancho de banda promedio obtenido (153 Mbit/sec):

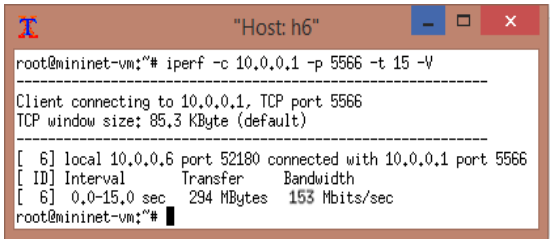


Figura 4.22. Imagen de los resultados prueba IPERF en IPv6 en el lado del cliente o host 6 “h6”.

También se realizó una prueba de conectividad, la cual se observa en la siguiente imagen (Figura 4.23) y en donde se obtuvo el delay de ocho paquetes con lo cual se puede obtener el promedio, el cual es 0.438625 ms.

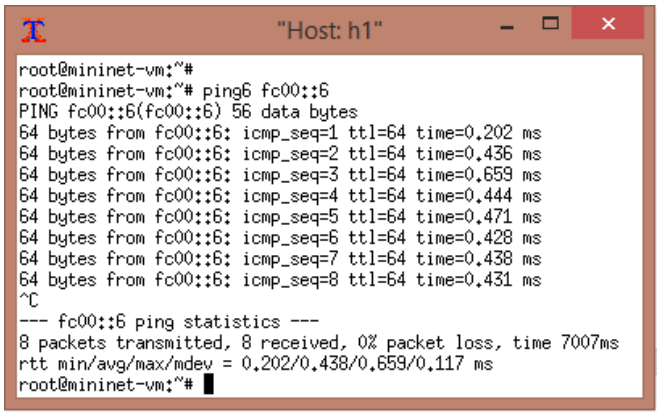


Figura 4.23. Imagen de la prueba de conectividad en IPv6 entre el “host 1” y el “host 6”.

De esta prueba obtenemos el siguiente gráfico (Figura 4.24) que nos muestra el desempeño:

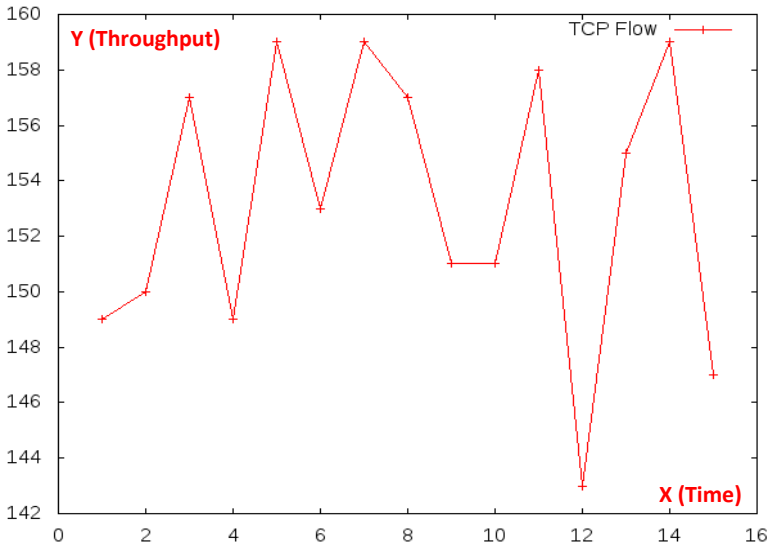


Figura 4.24. Imagen de la respuesta en throughput vs tiempo para la simulación de red de laboratorio SDN, caso 5 usando la herramienta IPERF en IPv6 y el editor gráfico GNUPLOT.

De esta imagen comparándola con la 4.11 podemos observar que en nuestro entorno simulado se alcanza un bit rate mayor en IPv6 para las mismas condiciones que en IPv4.

Se realizó el mismo ejercicio que en el caso 1, y se calculó el windows size óptimo el cual para este caso es: 56.144 KBps el cual es el windows size óptimo para este caso en IPv6.

•**Caso 6**: Prueba con configuración de **TCP** windows size 56 KB e IPv6, en la siguiente imagen (Figura 4.25) se observa los resultados en el servidor IPERF o host 1 “h1”.

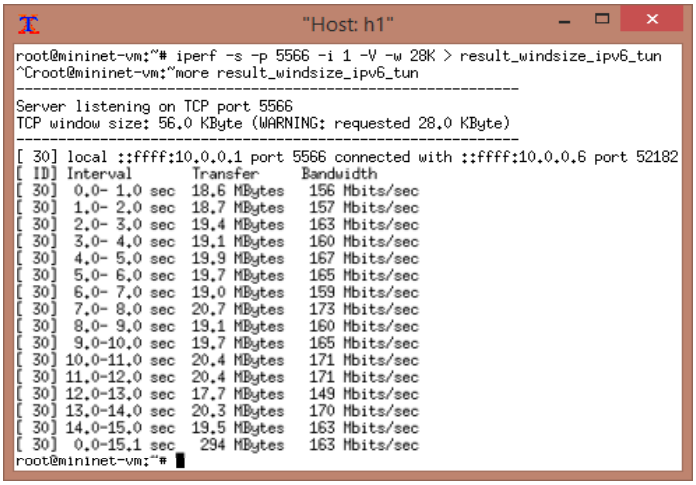


Figura 4.25. Imagen de los resultados prueba TCP con IPERF en IPv6 en el lado del servidor o host 1 “h1” considerando un windows size de 56 KB.

Como podemos observar la capacidad de throughput es ligeramente superior, siendo 163 Mbps respecto del caso anterior que fue 153 Mbps. En la siguiente imagen (Figura 4.26) se observa el desempeño:

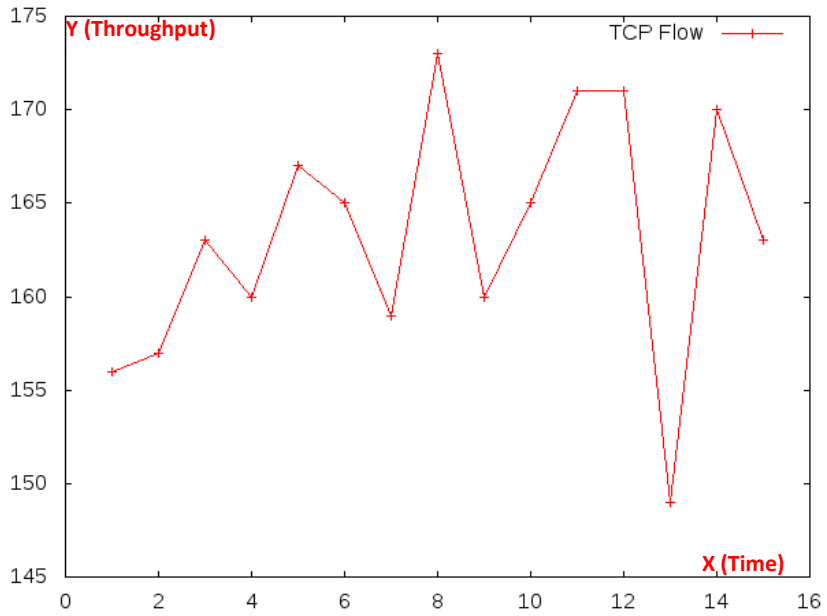


Figura 4.26. Imagen de la respuesta en throughput vs tiempo para la simulación de red de laboratorio SDN, caso 6 usando la herramienta IPERF en IPv6 y el editor gráfico GNUPLOT.

De esta imagen observamos que los picos mínimos y máximos son superiores respecto habiendo reajustado el windows size al tamaño óptimo.

•<u>Caso 7</u>: Prueba en **UDP** en **IPv6**, con un buffer size por defecto y una configuración de tasa de envío de **1Gbps**, en la siguiente imagen (Figura 4.27) se observan los resultados del lado del cliente, es decir, el host “h6”:

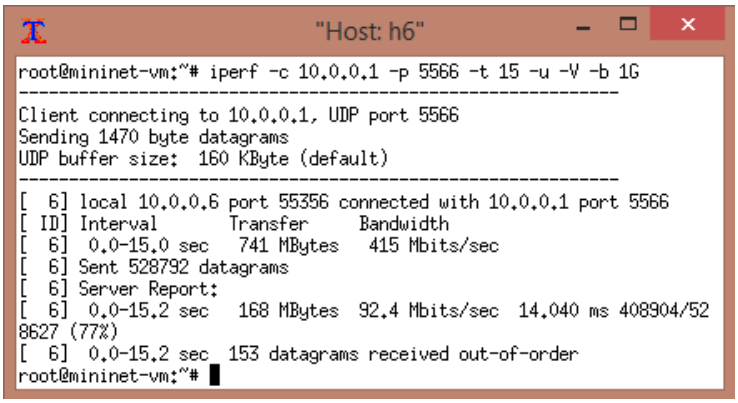


Figura 4.27. Imagen de los resultados prueba UDP con IPERF en IPv6 en el lado del cliente, usando una tasa de envío de 1Gbps.

En la siguiente imagen (Figura 4.28) se observan los resultados de manera detallada, del lado del servidor, es decir, host “h1”:

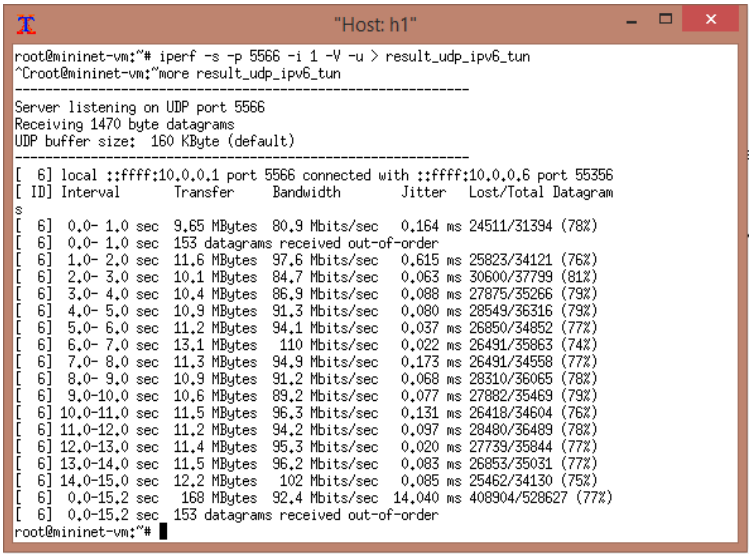


Figura 4.28. Imagen de los resultados prueba UDP con IPERF en IPv6 en el lado del servidor, usando una tasa de envío de 1Gbps.

Sin embargo, se observa que la herramienta se limita a 415 Mbits/sec y se pierden paquetes, por lo cual se analizará un nuevo caso dentro del desempeño soportado por la red.

Además, de esta simulación, se obtiene el siguiente gráfico (Figura 4.29) de desempeño:

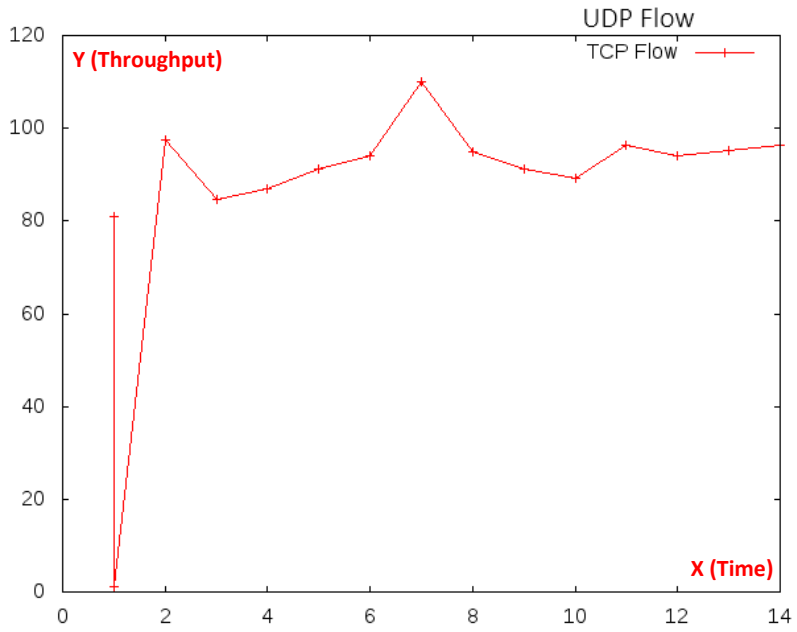


Figura 4.29. Imagen de la respuesta en throughput vs tiempo para la simulación de red de laboratorio SDN, caso 7 considerando una tasa de envío de 1Gbps.

De esta imagen podemos observar que existe un pico en el segundo 7 el cual llega a 110 Mbps, sin embargo, el resto de valores se encuentra alrededor de los 90 Mbps y es irregular.

•Caso 8: Prueba en **UDP** con **IPv6**, un buffer size por defecto y una configuración de tasa de envío de **100 Mbps (dentro de lo soportado)**, en la siguiente imagen (Figura 4.30) se puede observar los valores obtenidos a detalle:

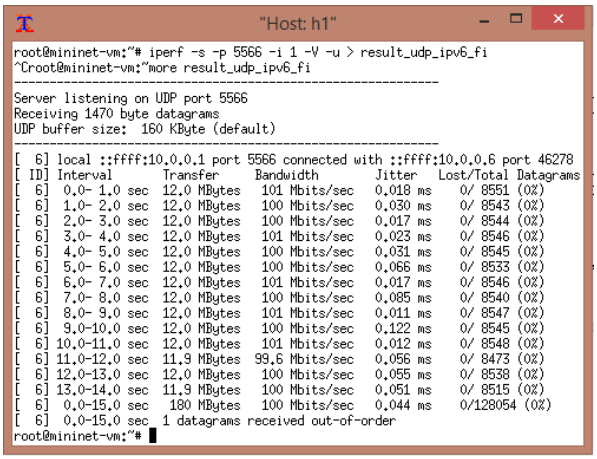


Figura 4.30. Imagen de los resultados prueba UDP con IPERF en IPv6 en el lado del servidor, usando una tasa de envío de 100 Mbps.

De esta prueba observamos que tenemos una pérdida de paquetes nula (0%) y el gráfico de desempeño se observa en la siguiente imagen (Figura 4.31):

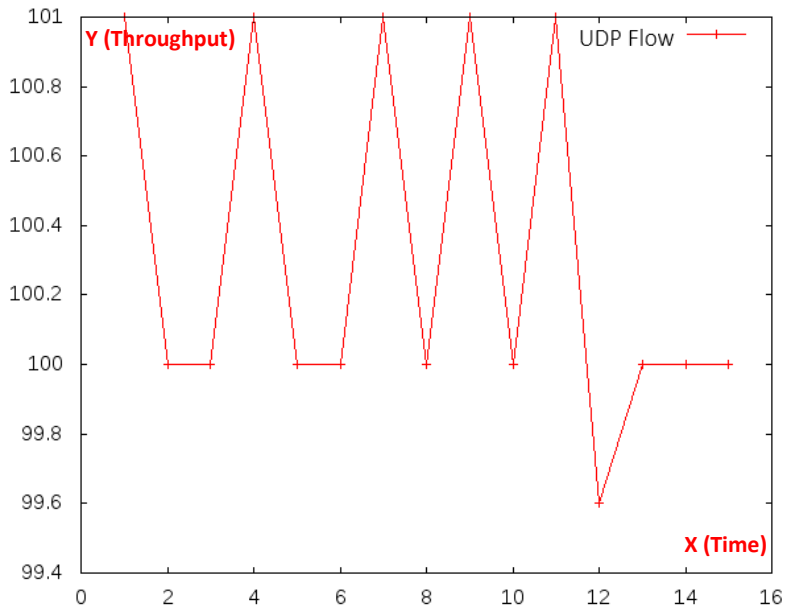


Figura 4.31. Imagen de la respuesta en throughput vs tiempo para la simulación de red de laboratorio SDN, caso 8 usando la herramienta IPERF y el editor gráfico GNUPLOT.

En esta gráfica observamos un resultado prácticamente constante ya que se encuentra entre 100 y 101 Mbps, esto debido a que la tasa de envío configurada es adecuada para la prueba en contraste con el caso 7.

4.3 RESULTADOS DE LAS SIMULACIONES

Luego de haber realizado los escenarios “A” y “B” y comparar sus resultados y gráficos de desempeño se concluye lo siguiente:

Para la reconvergencia de la red en IPv4 se tiene un tiempo aproximado de 5 segundos (desde “h1” hasta “h6”) en la simulación, lo cual sería interesante comparar con una red de laboratorio en la FIEE-UNMSM ya implementada, este tiempo es similar haciéndose la prueba tanto a los hosts 3 y 4 que implican un salto como para los hosts 5 y 6 teniéndose como referencia el host 1.

Se encontró un throughput mayor en IPv6 para TCP (respecto de TCP en IPv4), y a nivel de Protocolos IP (versión 4 y 6) se encontró que al cambiar el windows size en TCP al valor más adecuado (en IPv4 es 33.3 KB y en IPv6 56 KB) se puede incrementar el throughput alcanzado.

En UDP, se debe variar el parámetro bandwidth (para IPv4 e IPv6) a un valor alto para encontrar el valor limitante y luego hacer pruebas con este para luego escoger un valor dentro de lo soportado que lleve la prueba al límite, optimizando los resultados del throughput y observar el valor real.

En el siguiente gráfico (Figura 4.32) se observa la comparación en el throughout obtenido en la simulación, tanto para IPv4/IPv6 como para TCP/UDP.

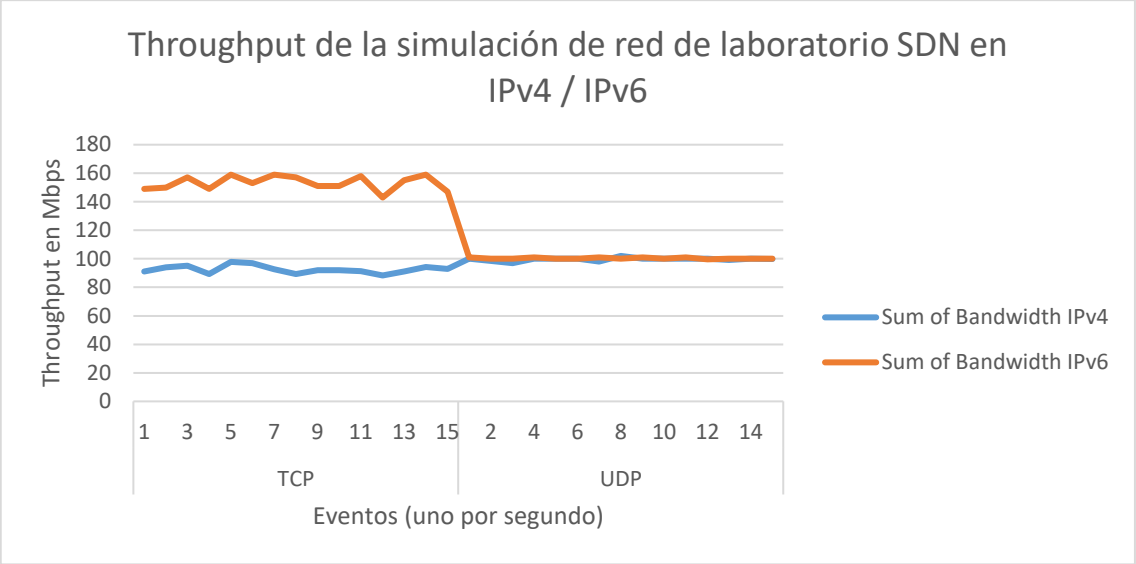


Figura 4.32. Gráfico throughput para distintas pruebas de la red de laboratorio propuesta, simulada en Mininet y haciendo uso de la herramienta IPERF.

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

6.1 CONCLUSIONES

a.- Como consecuencia del análisis de las especificaciones técnicas que definen la arquitectura de una Red Definida por Software se ha propuesto y simulado el funcionamiento y comportamiento con tráfico IPv4 e IPv6 de una red SDN para la Facultad de Ingeniería Electrónica y Eléctrica de la Universidad Nacional Mayor de San Marcos que facilitará la enseñanza de los alumnos y les permitirá afrontar los retos del ámbito profesional conociendo la tecnología a la cual se están dirigiendo las redes de datos de las compañías operadoras más importantes del país.

b.- Las simulaciones de la red SDN propuesta fueron realizadas usando la herramienta Mininet tanto para escenarios en IPv4 como IPv6: observándose que IPv6 es un 76.216 % superior en throughput para su prueba TCP con ventana optimizada (caso 6, figura 4.25, valor promedio 163 mbps) respecto al mismo parámetro en IPv4 (caso 2, figura 4.14, valor promedio 92.5 mbps)

c.- De los tres controladores usados HP VAN SDN, OpenDayLight y FloodLight, se seleccionó el OpenDayLight ya que ofrece más flexibilidad, posee mayor documentación y es open source, teniendo más módulos desarrollados que el FloodLight.

d.- En estos complicados momentos por la pandemia, es de vital importancia considerar la conexión remota al laboratorio tanto para los docentes como los alumnos.

e.- Es esencial que se empiecen a desarrollar aplicaciones en SDN en investigaciones posteriores, teniendo como base esta tesis.

6.3 RECOMENDACIONES

a.- Se recomienda implementar las siguientes prácticas de laboratorio entre las que existan en el curso Arquitectura de Redes Virtuales (SDN/NFV):

1) Reconocimiento del controlador, módulos, características e interfaces.

2) Integración del controlador con los nodos del plano de datos y pruebas de conectividad.

3) Implementación de APIs para la configuración de los equipos de red.

4) Implementación de APIs para la configuración de protocolos de red.

5) Medición y comparativa de rendimientos en redes legacy vs SDN.

b.- Considerando que la malla curricular 2018 ya se encuentra aprobada con el curso Arquitectura de redes virtuales (SDN/NFV) y que las empresas de telecomunicaciones del Perú ya han hecho pruebas con redes definidas por software y se encuentran en proceso de migración, se recomienda que se realice en la FIEE-UNMSM las coordinaciones pertinentes para la implementación de esta propuesta de laboratorio, que permitirá fortalecer las capacidades teóricas y técnicas de los egresados de esta facultad.

c.- Considerando que las SDN, ofrecen una gran flexibilidad para su gestión basándose en aplicaciones, se recomienda que futuros trabajos de investigación generen tesis en diseño y desarrollo de aplicaciones northbound en SDN.

d.- Considerando el desarrollo tecnológico actual no sólo implica SDN sino también virtualización y computación en la nube, se recomienda trabajar con servidores OpenStack y host virtualizados con un hipervisor de software libre como XEN CLOUD PLATFORM o KVM (Kernel Virtual Machine).

6.4 MATRIZ DE CONSISTENCIA:

MATRIZ DE CONSISTENCIA			
TITULO: Diseño y Simulación de una Red Definida por Software para la implementación de un laboratorio avanzado de datos para la EP de Telecomunicaciones de la Facultad de Ingeniería Electrónica y Eléctrica de la Universidad Nacional Mayor de San Marcos			
PROBLEMAS	**OBJETIVOS**	**HIPÓTESIS**	**VARIABLES**
Problema General	**Objetivo General**	**Hipótesis General**	**Variable Independiente**
¿Se puede realizar el diseño y simulación de una red SDN para educación e investigación en la FIEE-UNMSM con las herramientas de software libre disponibles?	Diseñar y Simular una Red Definida por Software para proponer la implementación de un laboratorio avanzado de datos en la Facultad de Ingeniería Electrónica y Eléctrica de la Universidad Nacional Mayor de San Marcos probando que ésta es teóricamente más eficiente en gestión y funcionamiento respecto a la legacy	Con las herramientas de software libre disponibles, se puede realizar el diseño y simulación de una red SDN para educación e investigación en la FIEE-UNMSM probando que es más eficiente en gestión y funcionamiento respecto a una red legacy.	- Diseño y simulación de una Red Definida por Sfotware para la implementación de un laboratorio avanzado de datos para la EP de Telecomunicaciónes de la FIEE-UNMSM
Problemas Específicos	**Objetivos Específicos**	**Hipótesis Específicas**	**Variable Dependiente**
1) ¿Es factible diseñar una topología de laboratorio SDN para la FIEE-UNMSM a nivel físico y lógico analizando las especificaciones técnicas de los principales organismos de estandarización que definen la arquitectura SDN? 2) ¿Es factible simular el funcionamiento del controlador open-source que se elegirá en el diseño utilizando contenedores o máquinas virtuales de software libre? 3) ¿Es factible simular el funcionamiento de la topología de red propuesta de un laboratorio SDN para la Facultad Ingeniería Electrónica y Eléctrica de la Universidad Nacional Mayor de San Marcos utilizando herramientas de software libre? 4) ¿Es factible diseñar la topología complementaria que permitiría el acceso remoto, habiendo hecho el diseño y simulación de la red de laboratorio SDN para la Facultad Ingeniería Electrónica y Eléctrica de la Universidad Nacional Mayor de San Marcos?	1) Diseñar la topología de laboratorio SDN para la FIEE-UNMSM, incluyendo la descripción de los equipos que debe tener. 2) Simular el funcionamiento del controlador open-source elegido para la red de laboratorio SDN. 3) Simular el funcionamiento de la topología propuesta de laboratorio SDN (controlador + nodos), utilizando herramientas de software libre, para la Facultad Ingeniería Electrónica y Eléctrica de la Universidad Nacional Mayor de San Marcos y revisar los resultados para analizar su mejora en eficiencia. 4) Diseñar la topología que permita conectividad remota al laboratorio SDN para la FIEE-UNMSM.	H1: Diseñando la topología de laboratorio SDN para la FIEE-UNMSM a nivel físico y lógico se probrarán los supuestos planteados para el correcto funcionamiento de acuerdo a los requerimientos del presente proyecto. H2: Utilizando contenedores o máquinas virtuales de software libre se puede simular el funcionamiento del controlador open-source que se elegirá en el diseño. H3: Utilizando herramientas de software libre se puede simular el funcionamiento de la topología de red propuesta de un laboratorio SDN para la Facultad Ingeniería Electrónica y Eléctrica de la Universidad Nacional Mayor de San Marcos comprobando que su tiempo de convergencia es menor respecto a una red legacy. H4: Habiendo hecho el diseño y simulación de la red de laboratorio SDN para la Facultad Ingeniería Electrónica y Eléctrica de la Universidad Nacional Mayor de San Marcos se puede diseñar la topología complementaria que permitiría el acceso remoto.	- Prácticas de laboratorio a realizarse en el curso Arquitectura de Redes Virtuales SDN/NFV.

BIBLIOGRAFÍA

[1] Fuente: Resolución Rectoral:

https://sum.unmsm.edu.pe/loginWebSum/Documento.pdf?nombreruta=19/07054-17t&nombredoc=07054-17t

[2] Fuente: OMS: https://www.paho.org/es/noticias/30-1-2020-oms-declara-que-nuevo-brote-coronavirus-es-emergencia-salud-publica-importancia

[3] Fuente: Documento Remote Learning and COVID-19 del banco mundial. Enlace:http://pubdocs.worldbank.org/en/925611587160522864/KnoweldgePack-COVID19-RemoteLearning-LowResource-EdTech.pdf

[4] Comunicado de entrega de dispositivos informáticos UNMSM. Enlace: https://viceacademico.unmsm.edu.pe/?news=comunicado-de-entrega-de-dispositivos-informaticos

[5] Decreto Legislativo N° 1465-2020. Enlace: https://cdn.www.gob.pe/uploads/document/file/605862/DL_1465.pdf

[6] Fuente: Ley 30309. Enlace: http://www.elperuano.com.pe/NormasElperuano/2015/03/13/1211074-1.html

[7] Fuente: Ley 28303. Enlace: http://www.pcm.gob.pe/wp-content/uploads/2016/06/Ley_28303_Ley_Marco_Ciencia_Tec_Innovacion_Tecnol%C3%B3gica.pdf

[8] Fuente: http://www.cip.org.pe/reconocimiento-del-peru-en-apec-engineer/

[9] Fuente: Web Oficial de Cisco.

Enlace: https://www.cisco.com/c/en/us/solutions/service-provider/software-defined-networks-sdn-service-providers/index.html#~stickynav=2

[10] Fuente: Web Oficial de Telefónica. Enlace: https://www.telefonica.com/en/web/press-office/-/telefonica-huawei-and-upm-perform-a-grounbreaking-field-trial-applying-quantum-cryptography-on-commercial-optical-networks-to-provide-secure-communica

[11] Fuente Definición de SDN en la web oficial de la Open Networking Foundation. Enlace: https://www.opennetworking.org/sdn-resources/sdn-definition

[12] Fuente: https://www.cisco.com/c/dam/m/en_us/solutions/enterprise-networks/networking-report/files/GLBL-ENG_NB-06_0_NA_RPT_PDF_MOFU-no-NetworkingTrendsReport-NB_rpten018612_5.pdf

[13] Fuente: Publicación: “Remote Learning and COVID-19” del Banco mundial. Enlace: http://pubdocs.worldbank.org/en/925611587160522864/KnoweldgePack-COVID19-RemoteLearning-LowResource-EdTech.pdf

[14]Fuente: Publicación de la EAIE (European Higher Education Area). Enlace: https://www.eaie.org/our-resources/library/publication/Research-and-trends/Coping-with-COVID-19--International-higher-education-in-Europe.html

[15] https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[16]Fuente: Trabajo de Fin de Grado para obtener el título de ingeniero informático en la Universidad Politécnica de Madrid “Despliegue de los laboratorios de SDN virtuales utilizando Open vSwitch” de Pablo Sayans Cobos de la Universidad Politécnica de Madrid (2018)

[17] Fuente: Artículo “Comparative analysis of SDN and conventional networks using routing protocols” de Deepthi Gopi, Samuel Cheng y Robert Huck (2017, USA)

[18]Fuente: Artículo “Performance analysis of SDN/OpenFlow controllers: pox versus floodlight” de Idris Z. Bholebawa y Upena D. Dalal (2017, España)

[19] Fuente: Artículo “Dynamic QOS/QOE assurance in realistic nfv-enabled 5g access networks” de Jose Juan Pedreno Manresa, Pouria Sayyad Khodashenas, Muhammad Shuaib Siddiqui y Pablo Pavon-Marino (2017, India)[13]

[20] Fuente: Artículo “Centflow: centrality-based flow balancing and traffic distribution for higher network utilization” de Rajesh Challa, Seil Jeon, Dongsoo S.Kim y Hyunseung Choo (2017, USA)

[21] Fuente: Artículo “Making powerful friends: introducing onos and net2plan to each other” de Pontus Sköldström Ćiril Rožić, Jose Juan Pedreno Manresa (2017, España)

[22] Fuente: Artículo “NFV and SDN guide for carriers and service providers” (2017)

[23] Fuente: Tesis de maestría: “Trusted communication in SDN openflow channel” de Marika Haapajärvi de la JAMK University of Applied Sciences (2017, Finlandia)

[24] Fuente: En la Tesis de grado “Prototipo de una SDN utilizando herramientas open-source” presentada por Juan Francisco Guano Viscarra de la Escuela Politécnica Nacional de Quito (2017, Ecuador)

[25] Fuente: Tesis de Maestría “Analysis of openflow and netconf as sbis in managing the optical link interconnecting data centers in an SDN environment” presentado por Karpakamurthy Muthukumar de la Universidad de Carleton, Canadá (2016)

[26] Fuente: Tesis “Diseño e implementación de un controlador SDN/OpenFlow para una red de campus académica” presentada por los bachilleres Gabriel Josías Cuba Espinoza y Juan Manuel Augusto Becerra Avila de la PUCP (2016)

[27] Fuente: Tesis de fin de master “Estudio de las redes definidas por software y escenarios virtuales de red orientados al aprendizaje” de Javier Cano Moreno de la Universidad Politécnica de Madrid (2015)

[28] Fuente: Tesis fin de grado (TFG) “Estudio de la factibilidad técnica en la implementación de la tecnología SDN en las redes de datos de área local” de Xavier Andrés Domínguez Briones. (2015 Universidad Politécnica Salesiana)

[29] Fuente: Tesis de Grado “Opendaylight SDN controller platform” de Bernat Ribes García de la Universidad Politécnica de Catalunya, España (2015)

[30] Fuente: Trabajo de fin de master “Estudio del estado del arte de la ingeniería de tráfico en redes SDN. caso de estudio Oshi” de María José Argüello Vélez. (2015)

[31] Fuente: White paper “Practical implementation of SDN & NFV in the WAN”, octubre (2013)

[32] Fuente: RFC 7426 Páginas 12-16

[33] Fuente: Software Defined Networks: A Comprehensive Approach de Paul Goransson y Chuck Black Páginas 71-75

[34] Fuente: Software Defined Networks: A Comprehensive Approach de Paul Goransson y Chuck Black Página 91

[35] Fuente: Software Defined Networks: A Comprehensive Approach de Paul Goransson y Chuck Black Página 94

[36] Fuente: Software Defined Networks: A Comprehensive Approach de Paul Goransson y Chuck Black Página 93

[37] Fuente: RFC 6241

[38] Fuente: OpenFlow Switch Specification versión 1.3.5 Pag 12

[39] Fuente: OpenFlow Switch Specification versión 1.3.5 Pag 12

[40] Fuente: OpenFlow Switch Specification versión 1.3.5 Pag 12

[41] Fuente: Open Networking Foundation. Enlace: https://www.opennetworking.org/sdn-resources/sdn-definition

[42] Fuente: Software Defined Networks: A Comprehensive Approach de Paul Goransson y Chuck Black Página 61-62

[43] Fuente: Software Defined Networks: A Comprehensive Approach de Paul Goransson y Chuck Black Página 61-62

[44] Fuente: Software Defined Networks: A Comprehensive Approach de Paul Goransson y Chuck Black Página 61-62

[45] Fuente: OpenFlow Switch Specification versión 1.3.5 Pag 9

[46] Fuente: OpenFlow Switch Specification versión 1.3.5 Pag 18,19

[47] Fuente: OpenFlow Switch Specification versión 1.3.5 Pag 18,19

[48] Fuente: OpenFlow Switch Specification versión 1.3.5 Pag 18,19

[49] Fuente: OpenFlow Switch Specification versión 1.3.5 Pag 18,19

[50] Fuente: OpenFlow Switch Specification versión 1.3.5 Pag 18,19

[51] Fuente: OpenFlow Switch Specification versión 1.3.5 Pag 18,19

[52] Fuente: OpenFlow Switch Specification versión 1.3.5 Pag 21,22

[53] Fuente: OpenFlow Switch Specification versión 1.3.5 Pag 23

[54] Fuente: OpenFlow Switch Specification versión 1.3.5 Pag 23,24

[55] Fuente: OpenFlow Switch Specification versión 1.3.5 Pag 16-21

[56] Fuente: OpenFlow Switch Specification versión 1.3.5 Pag 31-33

[57] Fuente: OpenFlow Switch Specification versión 1.3.5 Pag 31-33

[58] Fuente: OpenFlow Switch Specification versión 1.3.5 Pag 31-33

[59] Fuente: OpenFlow Switch Specification versión 1.3.5 Pag 31-33

[60] Fuente: Artículo Automatic bootstrapping of OpenFlow networks de Sachin Sharma, Dimitri Staessens, Didier Colle, Mario Pickavet and Piet Demeester (2013)

[61] Fuente: Software Defined Networks: A Comprehensive Approach de Paul Goransson y Chuck Black Páginas 71-75

[62] Fuente: Interfaces, Attributes, and Use Cases: A Compass for SDN de Michael Jarschel, Thomas Zinner, Tobias Hoßfeld, Phuoc Tran-Gia, y Wolfgang Kellerer (2014)

[63] Fuente: https://www.opendaylight.org/what-we-do/odl-platform-overview

[64] Fuente: https://docs.opendaylight.org/en/stable-fluorine/developer-guide/yang-tools.html Yang Tools Overview

[65] Fuente: RFC 6020 YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)

[66] Fuente: OpenDayLight Summit: YANG User Interface (YANGUI) in OpenDaylight. Enlace: https://events.static.linuxfound.org/sites/events/files/slides/YANGUI-metz-malachovsky-sebin-ODL-Summit-final-July29.pdf

[67] Fuente: RFC 6020 YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)

[68] Fuente: https://developer.cisco.com/codeexchange/github/repo/CiscoDevNet/OpenDaylight-Openflow-App/

[69] Fuente: http://www.projectfloodlight.org/floodlight/

[70] Fuente: https://www.sdxcentral.com/sdn/definitions/sdn-controllers/opendaylight-controller/

[71] Fuente: https://www.sdxcentral.com/sdn/definitions/sdn-controllers/opendaylight-controller/

[72] Fuente: ONOS Overview Tenets, Roadmap, Deployments (March, 2018)

[73] Fuente: ONOS Overview Tenets, Roadmap, Deployments (March, 2018)

[74] Fuente: ONOS Overview Tenets, Roadmap, Deployments (March, 2018)

[75] Fuente: HPE VAN SDN Controller 2.6 Administrator Guide

[76] Fuente: HPE VAN SDN Controller 2.6 Administrator Guide

[77] Fuente: http://mininet.org/walkthrough/#interact-with-hosts-and-switches

[78] Fuente: OpenFlow Switch Specification versión 1.3.5 Pag 50

[79] Fuente: OpenFlow Switch Specification versión 1.3.5 Pag 119,120

[80] Fuente: OpenFlow Switch Specification versión 1.3.5

[81] Fuente: OpenFlow Switch Specification versión 1.3.5 Pag 87

[82] Fuente: OpenFlow Switch Specification versión 1.3.5 Pag 113

[83] Fuente: OpenFlow Switch Specification versión 1.3.5 Pag 79

[84] Fuente: *Architectural Styles and the Design of Network-based Software Architectures*. De Roy Thomas. Doctoral dissertation, University of California, Irvine, 2000.

[85] Fuente: Software Defined Networks: A Comprehensive Approach de Paul Goransson y Chuck Black, Páginas 307-308.

[86] Fuente: https://www.snia.org/about/corporate_info

ANEXO A

Detalles del funcionamiento del controlador HP VAN. A continuación, en la figura A.1 se observa el GUI del HPVAN.

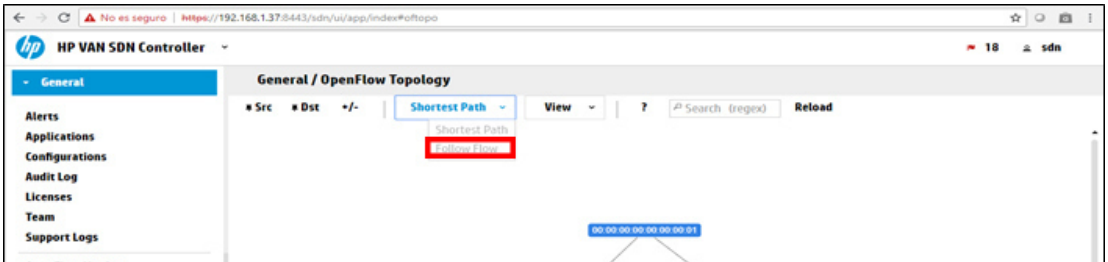


Figura A.1. Grafic User Interface de controlador HP Van SDN.

En Applications se tiene las diferentes aplicaciones que se ha instalado en el controlador (compradas previamente), por defecto vienen las de descubrimiento de nodos, enlaces y diagnóstico de ruta.

En Configurations se tiene las configuraciones aplicadas al controlador, figura A.2, como por ejemplo el hard/idle timeout de las tablas de flujo.

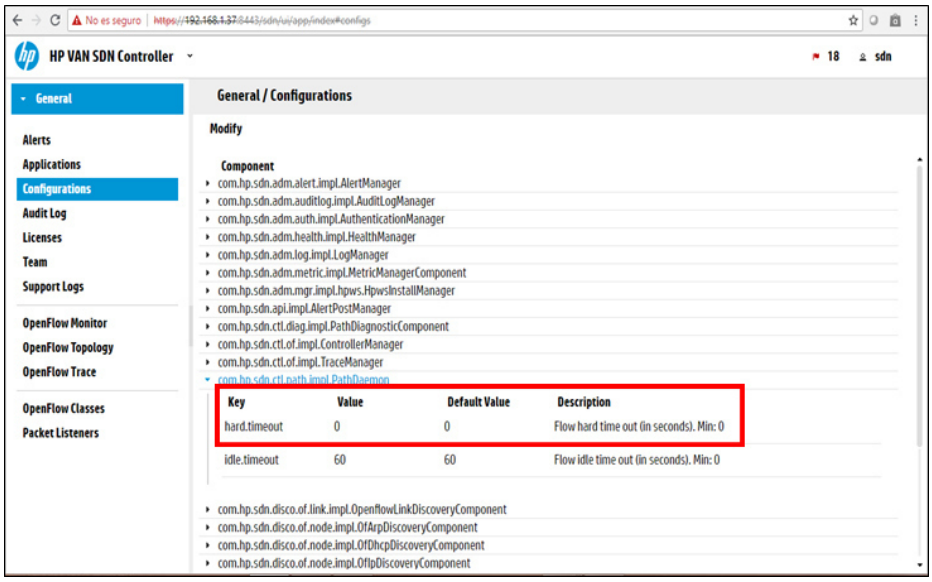


Figura A.2. Configuración del Idle/hard timeout en el controlador HPVAN SDN.

En Audit Log se encuentran los registros de las IPtables usadas desde que se inició el controlador.

En Licences están las licencias activadas en el controlador, se pueden agregar o quitar.

En Support Logs se tiene los registros genéricos, es decir los registros de los token al ingresar al controlador, cuando se eliminó o agregó tablas, reinicio de caché, etc.

En el OpenFlow Monitor, figura A.3, se muestra los nodos en la red, tanto la información del fabricante, número de serie, buffer, tablas, puertos, flujos, etc. También permite revisar las tablas de flujo enviadas a cada nodo, el primer flujo es del protocolo de descubrimiento de enlaces de HP, BDDP (la versión broadcast del protocolo LLDP), el segundo flujo es para el envío del protocolo ARP, el tercero para el manejo de los flujos perdidos mediante el reenvío legacy, el cuarto para el envío de mensajes DHCP.

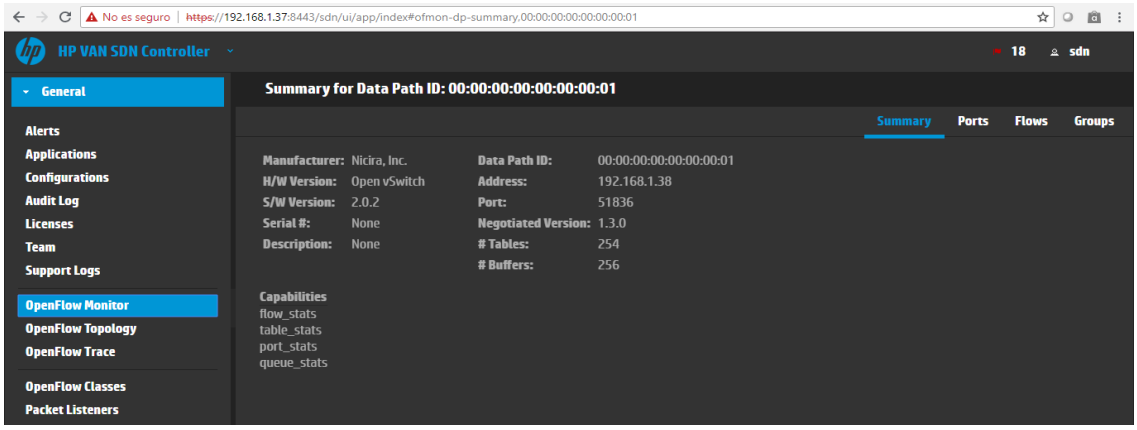


Figura A.3. Datos del OVSwitch detectado por el controlador HP VAN.

En la figura A.4 se puede observar en el apartado “Flows” las tablas de flujo gestionadas por el controlador.

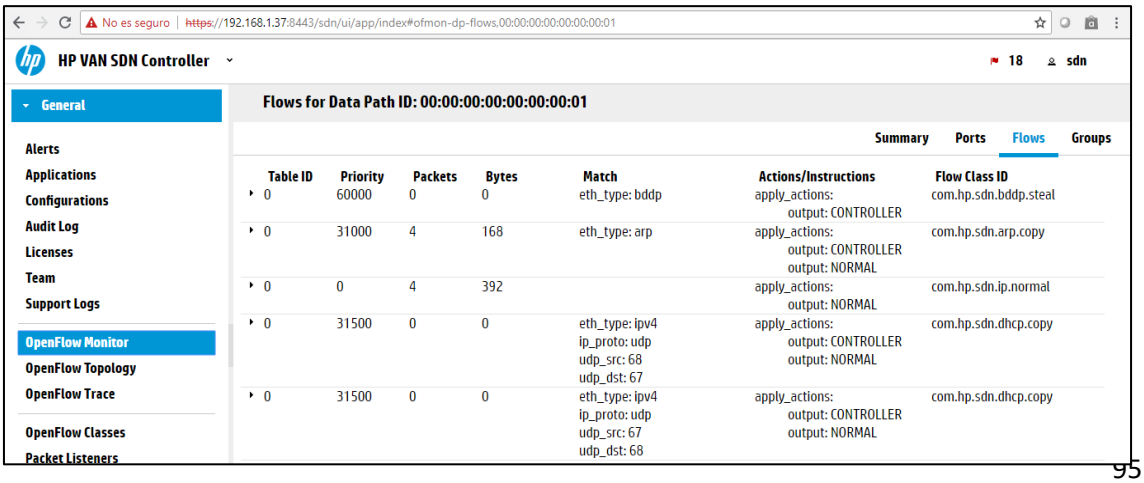


Figura A.4. Tablas de flujo enviadas del controlador HP Van al OVswitch.

En OpenFlow Topology se observa la topología manejada por el controlador, en OpenFlow Trace ilustrado en la figura A.5 se puede hacer el seguimiento a los mensajes y se muestra el Data Pad Id específico (nodo), durante el tiempo que deseemos.

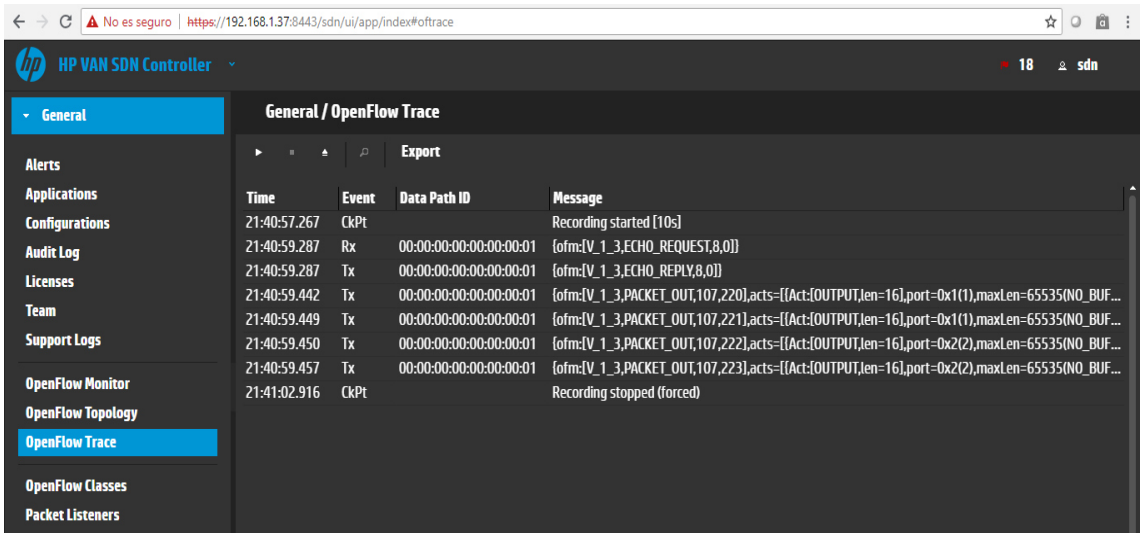


Figura A.5. Seguimiento a los mensajes gestionados por el OVSwitch.

La programación de aplicaciones es en lenguaje java y el envío de instrucciones se hace con la API Rest del controlador o usando el cliente Rest llamado Postman.

Si bien el controlador HP Van tiene más módulos, además de soporte, todo va ligado a licencias por lo cual es un gasto a considerar si se desea implementar.

ANEXO B

El controlador Floodlight, también es de desarrollo libre, licenciado por Apache y basado en java, aunque posee menos módulos y bibliografía con respecto al

OpenDayLight, en el Dashboard mostrado en la figura B.1 se observan los nodos detectados por el controlador, la información de sus puertos, paquetes, etc.

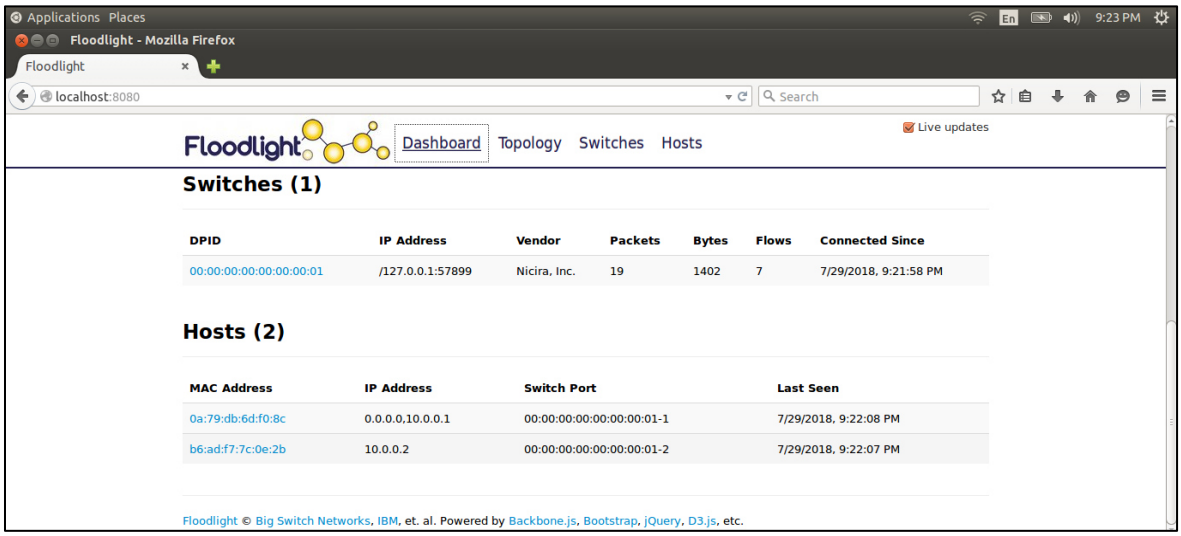


Figura B.1. Dashboard del controlador Floodlight.

En la figura B.2 se observa la topología detectada por el controlador y las Mac correspondientes.

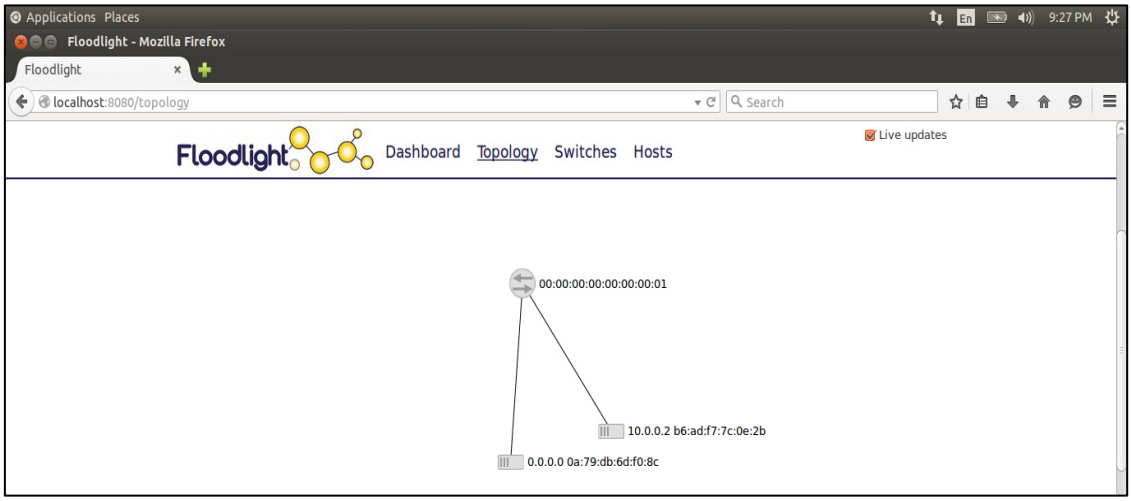


Figura B.2. Topología gestionada por el controlador Floodlight.

En la parte de Switches, figura B.3, se observan los Ovswitches, sus puertos, sus características de fábrica y los flujos gestionados.

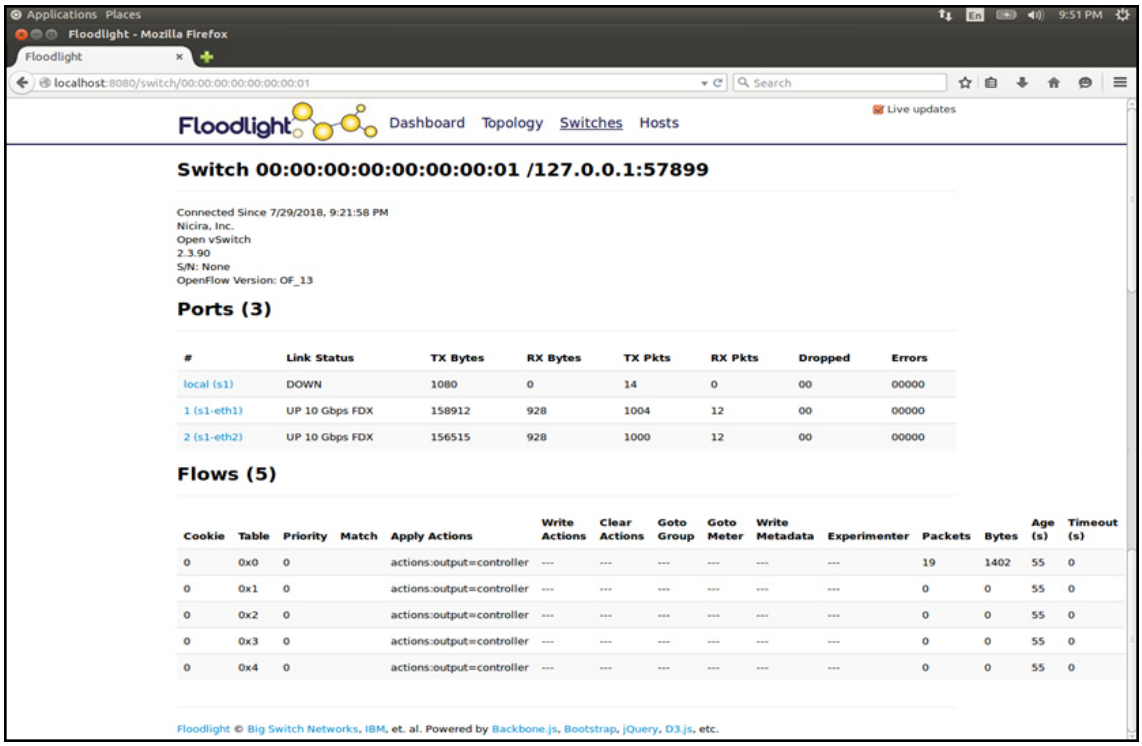


Figura B.3 Tablas de flujo gestionadas en Floodlight.

En el apartado de Hosts, figura B.4, están agrupados los clientes, tanto sus Mac, IP's, puertos a los que están conectados y la última hora de actividad.

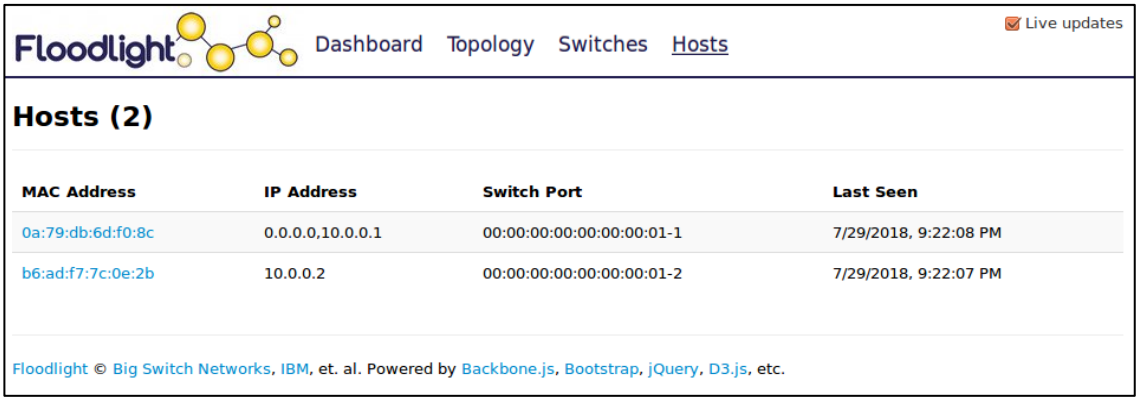


Figura B.4. Host descubiertos por el controlador Floodlight.

ANEXO C

El VM Mininet, trae por defecto desactivado el uso de IPv6, para activarlo se encontró la siguiente solución, haciendo cambios en la máquina Ubuntu que lo contiene, es decir, directo en el SO:

1) Ir al archivo sysctl.conf que se encuentra dentro del directorio /etc para ello escribir sudo vim /etc/sysctl.conf y buscar las líneas que dicen net.ipv6.conf.all.disable_ipv6 = 1, net.ipv6.conf.default.disable_ipv6 = 1, net.ipv6.conf.lo.disable_ipv6 = 1 y ponerlas como comentario añadiendo una almohadilla delante de ellas, es decir, quedarán de la siguiente manera:

#net.ipv6.conf.all.disable_ipv6 = 1, #net.ipv6.conf.default.disable_ipv6 = 1, #net.ipv6.conf.lo.disable_ipv6 = 1

2) Ir a la carpeta /etc/default/grub, buscar la línea: GRUB_CMDLINE_LINUX_DEFAULT="ipv6.disable=1 text" y borrar la parte ipv6.disable=1, de modo que quede:

   GRUB_CMDLINE_LINUX_DEFAULT="text" luego escribir el comando sudo update-grub y luego sudo reboot

ANEXO D

Para las conexiones SSL, se tienen que asignar las credenciales correspondientes del lado del firewall Fortinet, y del lado de los clientes se tiene que instalar el “Cliente VPN FortiClient” siguiendo los siguientes pasos:

1) Instalar el SW Forti Client:

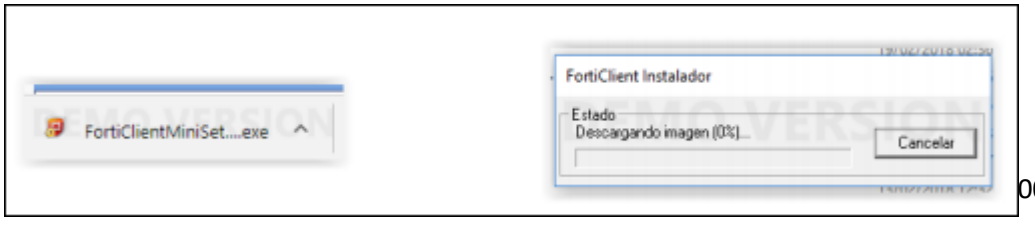


Figura D.1 Instalación FortiClient

2) Dar click a la casilla “Yes, I have read and accept the License Agreement”, y luego click en “Next”:

Figura D.2 FortiClient Setup Wizard

3) Seleccionar sólo la opción “VPN Only” y luego dar click la opción “Next”

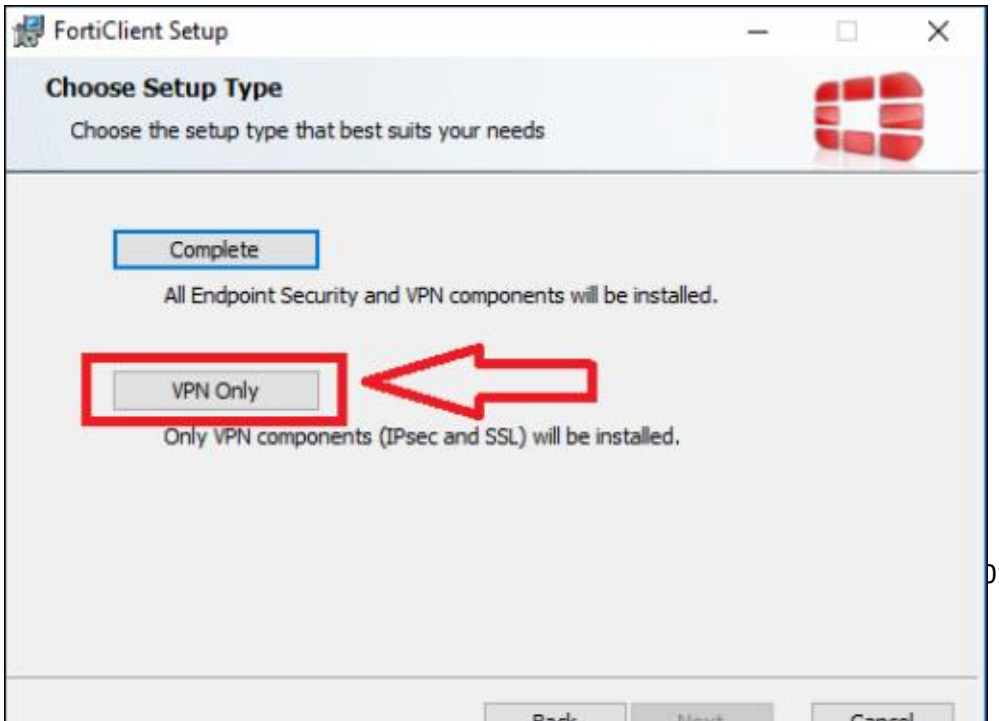


Figura D.3 FortiClient Setup Type

4) Selección de ruta de almacenamiento:

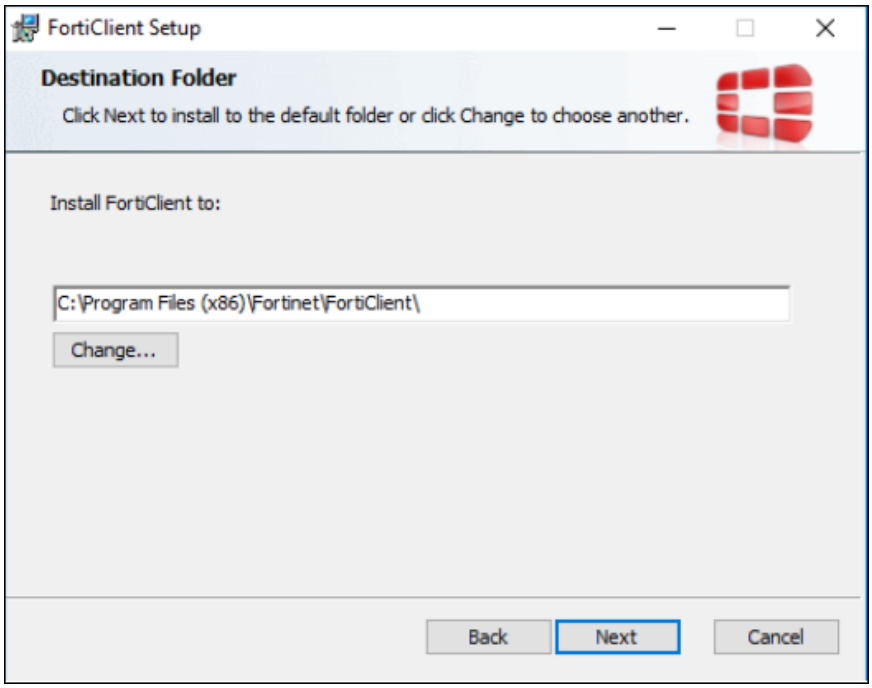
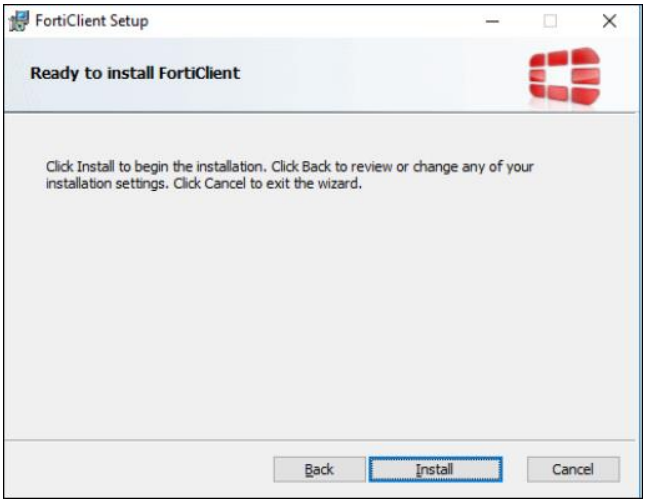


Figura D.4 FortiClient ruta de destino.

5) Culminación de la instalación:



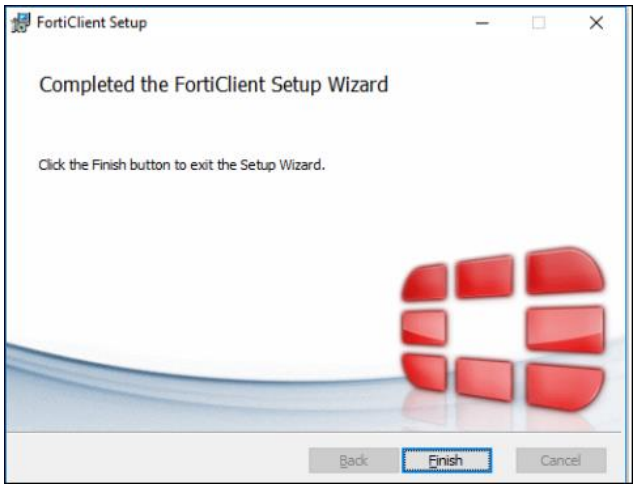


Figura D.5 FortiClient finalización de instalación.

6) Configuración de la VPN:

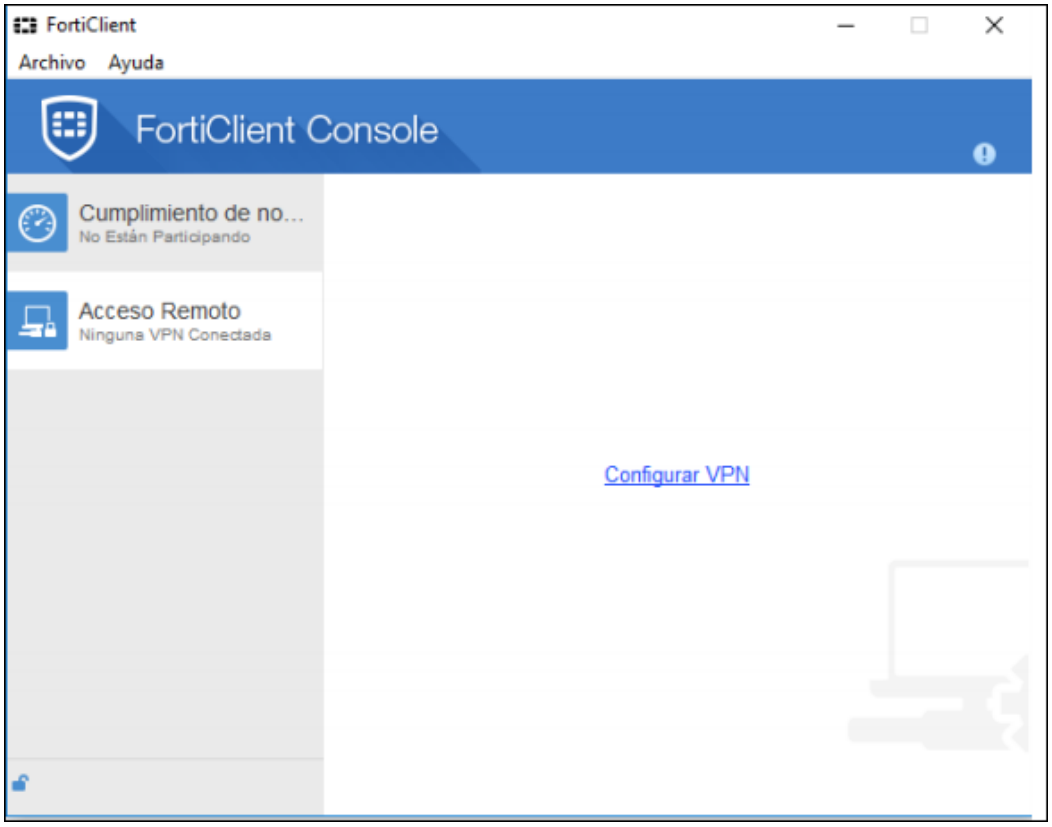


Figura D.6 FortiClient configuración de la VPN (a).

7) Colocar el Gateway Remoto, como ejemplo en nuestro caso “webvpnfiee-unmsm.com.pe”, así como el nombre de la conexión y la descripción los cuales son de libre elección, el puerto puede ser uno específico, por defecto es el 443 pero se puede usar uno en particular como el 55443 (configurado previamente en el Fortinet), luego dar en “aplicar” y por último en “cerrar”:

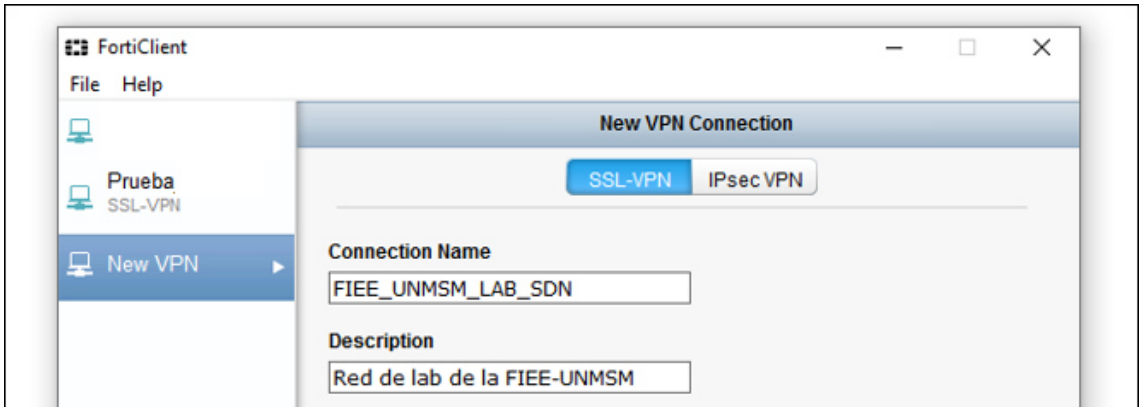


Figura D.7 FortiClient configuración de la VPN (b).

8) Colocar las credenciales de usuario y la contraseña y dar click en conectar:

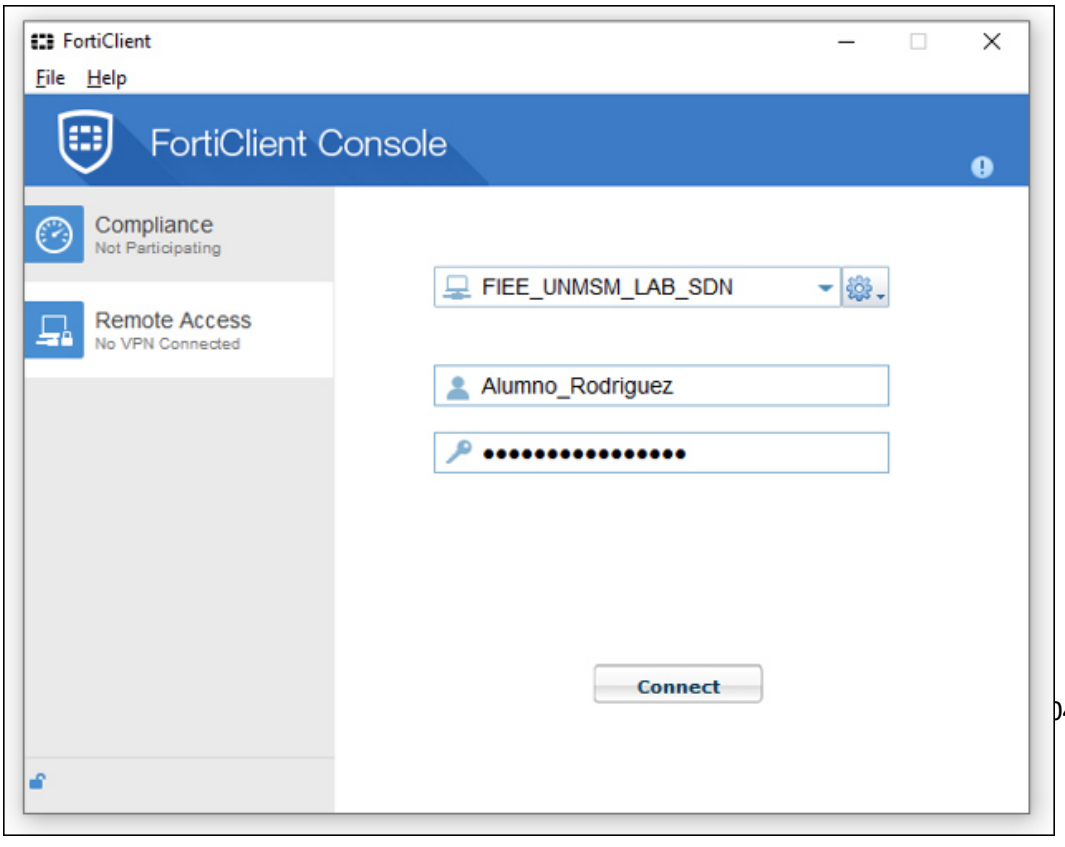


Figura D.8 FortiClient credenciales para la conexión.

9) Con esto se logra la conexión VPN SSL y la PC se encuentra en la red de laboratorio, como si estuviera conectada en la intranet:

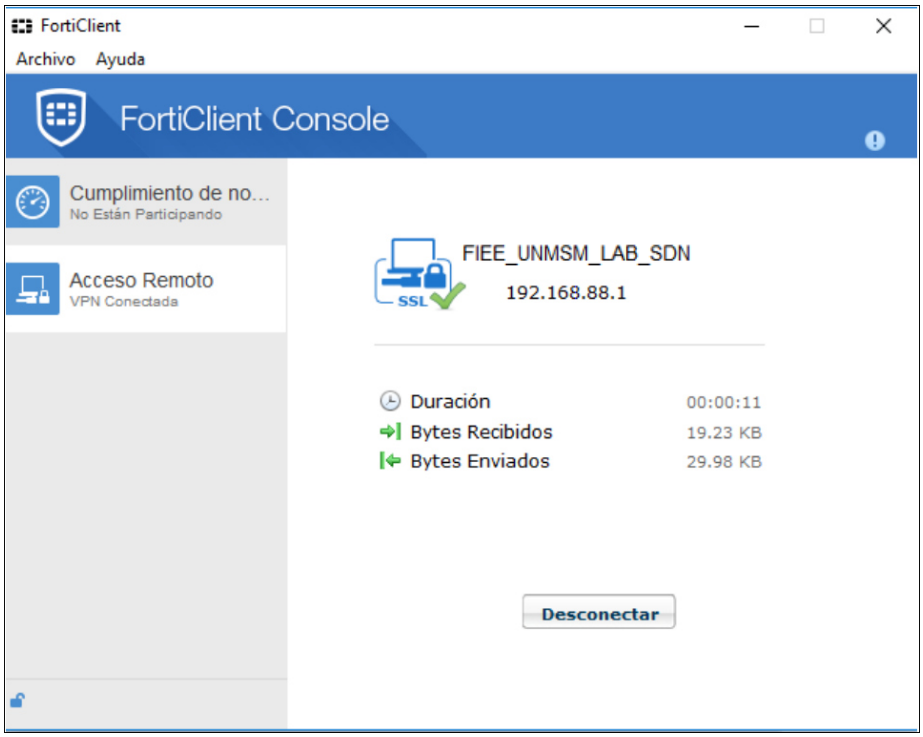


Figura D.9 FortiClient conexión VPN SSL activa.